Multilingual Techniques for Low Resource Automatic Speech Recognition

by

Ekapol Chuangsuwanich

Submitted to the
Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2016

Author .....
Department of Electrical Engineering and Computer Science
May 20, 2016

Certified by .....
Dr. James Glass
Senior Research Scientist
Thesis Supervisor

Accepted by .....
Professor Leslie A. Kolodziejski
Chair, Department Committee on Graduate Students

Multilingual Techniques for Low Resource Automatic Speech Recognition

by

Ekapol Chuangsuwanich

Submitted to the
Department of Electrical Engineering and Computer Science
on May 20, 2016, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Out of the approximately 7000 languages spoken around the world, there are only about 100 languages with Automatic Speech Recognition (ASR) capability. This is due to the fact that a vast amount of resources is required to build a speech recognizer. This often includes thousands of hours of transcribed speech data, a phonetic pronunciation dictionary or lexicon which spans all words in the language, and a text collection on the order of several million words. Moreover, ASR technologies usually require years of research in order to deal with the specific idiosyncrasies of each language. This makes building a speech recognizer on a language with few resources a daunting task.

In this thesis, we propose a universal ASR framework for transcription and keyword spotting (KWS) tasks that work on a variety of languages. We investigate methods to deal with the need of a pronunciation dictionary by using a Pronunciation Mixture Model that can learn from existing lexicons and acoustic data to generate pronunciation for new words. In the case when no dictionary is available, a graphemic lexicon provides comparable performance to the expert lexicon. To alleviate the need for text corpora, we investigate the use of subwords and web data which helps improve KWS spotting results. Finally, we reduce the need for speech recordings by using bottleneck (BN) features trained on multilingual corpora. We first propose the Low-rank Stacked Bottleneck architecture which improves ASR performance over previous state-of-the-art systems. We then investigate a method to select data from various languages that is most similar to the target language in a data-driven manner, which helps improve the effectiveness of the BN features. Using techniques described and proposed in this thesis, we are able to more than double the KWS performance for a low-resource language compared to using standard techniques geared towards rich resource domains.

Thesis Supervisor: Dr. James Glass
Title: Senior Research Scientist

Acknowledgments

It has been almost seven years since I started here at MIT, and almost thirteen years in the US. Along the journey to complete my PhD degree, I have encountered so many wonderful people who made the time here much more fulfilling.

First, I would like to thank my advisor Jim Glass. To me, Jim is both a friend and a mentor. As an advisor, he gives meaningful advice and support for my work here at MIT. He also gives me plenty of liberty to pursue any particular aspect of the work I would like to work on. As a friend, we chat about non work-related things while partying at his house, at the Muddy, or even during regular lunch. I also would like to thank my committee, Regina Barzilay and Victor Zue, who provide guidance for the thesis. Victor, with his linguistic and ASR expertise, and Regina, bringing in another point of view from the NLP side, really help shape the direction of some of the work in this thesis.

The work here at MIT has been enjoyable because of the people here, especially the SLS group which includes (students, visitors, and staff; past and present) Ann, Carrie, Chen, Daniel, Dave, Eann, Felix, Hassan, Hung-an, Hung-yi, Ian, Ibrahim, Jackie, Jennifer, Jingjing, Kevin, Lee, Leo, Mandy, Marcia, Michael, Mitch, Mitra, Najim, Patrick, Scott, Sean, Sree, Stephanie, Stephen, Timo, Tuka, Wei-Ning, William, Xue, Yaodong, Yonatan, Yu, and Yushi. My interactions with them are very fruitful, and their feedback really helps me and my research grow. My 32-G442 officemates (past and present), in particular, are really helpful ranging from the most trivial matters to research discussion on our always messy whiteboard (and now quoteboard). Hung-an who was a senior student when I first joined, really helped me get started with SUMMIT and other SLS tools. Yaodong helped start the SLSDBM toolkit which I used extensively in this thesis. Yu worked with me on Babel for many years and helped me in the many of the work that went into this thesis. Leo also helped with SLSDBM and many Babel-related infrastructures. Stephen (yes, you were here for a Summer and still pokes in everyday anyway), who joined MIT the same year and will be graduating the same time as me, really makes life in MIT enjoyable. As one

of the few native speaker in SLS, he also helped proofread all of my papers. I also would like to especially thank the SLS people whose work went directly into this thesis, including Ian with his PMM, Hung-yi who helped with KWS, and Dave who took over Ian's PMM and exchanged many fruitful conversations. The support and technical staff here are also very prompt and caring; Marcia who takes care of the mundane yet important matters, Lee who maintained SUMMIT while he was here, Najim who helped with the Babel project on feature normalization and shared various discussions in life and research, and Scott who I got to work closely with during the forklift project and various server related issues. My work here would be a lot tougher without them.

Next, I would like to thank the Babelon Team, which consist of BBN, BUT, JHU, LIMSI, and NWU. Our weekly calls encompass a lot of the work done in this thesis. I would like to especially thank Stavros for leading the team and facilitate many of the collaborations. With a project as large as Babel, I had to rely on many people along the way; Damianos for his help on KWS. Rich for his insight and suggestions, especially on the frame selection work. Le for the web data and scoring. Frantisek, Igor, Karel, and Martin from BUT who provide many infrastructure in Kaldi, various resources, and discussion about DNN training. Marelie and Charl for the syllabication and phone mapping. Lori for providing VAD segmentations, and various discussions. Karthik for the morph segmentations.

Over the years I am fortunate to be able to collaborate with many great people. First, I'd like to thank Professor Richard Stern, my advisor at CMU, who made me fall in love with the field which I still continue to work on to this day. My brief time with Seth Teller, who leaded the forklift project, as my first real world application project was enjoyable and eye-opening. I was able to work with people at NTT, such as Hori-san and Watanabe-san which solidifies many of my understanding about Bayesian models and ASR in general. I got a chance to work with Ian again during an internship at Google, where I got to apply some of my multilingual framework in real world applications. The people there, especially, Carolina, Guoguo, Rohit, and Tara really provides a lot feedback and suggestion both for the work there and the

work here in MIT. Guoguo, in particular, really helped me with KWS-related issues.

I also would like to thank my friends, family, and the Thai community in the Boston area, such as TSMIT and OSK New England, without their love and support. I would not have made it this far.

Last but not least, all this would not have happened without the generous support from my sponsors. The Thai government had been funding my education in the US for around a decade. Without them, especially the support staff in OEADC and OCSC, all this would not have been possible. Next, I would like to thank IARPA for providing the research funding and data. The data really helps boost the community's interest in low resource ASR research.

This work was supported in part by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Defense US Army Research Laboratory contract number W911NF-12-C-0013. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoD/ARL, or the U.S. Government.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Automatic Speech Recognition (ASR) research has grown significantly over the past decade. With the emerging trend of smart devices and big data, companies and government agencies have invested heavily in speech technology. The 'naturalness' of spoken interfaces helps alleviate the need for physical input interfaces. Collections of video and audio data, such as on Youtube, need ASR capability for efficient search and other downstream processing such as translation and summarization.

However, amongst the approximately 7000 languages spoken around the world, there are only around 100 languages with speech recognition capability [66]. This is because to create a reliable recognition system requires a large amount of annotated data and linguistic knowledge. The standard recipe for building a speech recognizer typically requires: (1) thousands of hours of transcribed speech for the statistical learning of the acoustic model, (2) a phonetic pronunciation dictionary which determines how words of the language are decomposed into smaller phone-like units, and (3) a large collection of text to create the language model. Because of the expensive process to generate these three language resources, most of the research in ASR techniques has been predominately focused on English and a few other major languages. Moreover, even with the aforementioned resources, because of language specific traits, it typically requires several months to years of research effort to push the performance on any particular language to be in line with the languages which researchers have been working on for decades. Thus, the time from the point where the need for ASR

on any particular language arises, to the time of system deployment can take years, which in many cases might be too late for the target application.

To address the data limitations, we must come up with a training framework that requires less annotated resources, is language independent to minimize human experts, and can be deployed in a reasonable time frame. In this thesis, we propose a method that can better identify and utilize the similarity in acoustics between languages to introduce data sharing from rich resource languages in a multilingual manner to alleviate the acoustic data requirements. Since the underlying structure that humans use to produce speech sounds are the same across languages, we expect the models learned in this manner to have properties that can be transferred to use in languages with limited resource.

1.1 Effect of training data size on ASR performance

To better illustrate how lack of training data can affect ASR performance, Figure 1-1 shows the Word Error Rate (WER) as a function of training data (in hours) for various telephone conversational speech transcription tasks. It is estimated that human performance on this task is around 4% WER [51]. For ASR, there are experiments conducted by IBM [78] and Microsoft [82] using around 2000 hours transcribed data, with a WER as low as 8%, doubling human performance. Then, there are ASR systems trained on the Switchboard corpus, using 300 hours of training data, that perform between the 13-20% WER range [84, 86]. However, on non-English tasks, the amount of available training data can be a lot smaller, since transcribing 1 minute of spontaneous speech can take 7-9 minutes to transcribe properly [2]. The WER goes up to 35% with 150 hours on training data on Mandarin [50] and Spanish [46]. With 10 hours of training data, around half of the hypothesized words will be wrong. With 3 hours of training data, in some languages such as Kurmanji and Telugu, the WER can shoot up above 80%. Above 80% WER, ASR systems are no longer reliable, even

for simpler downstream tasks such as keyword spotting.

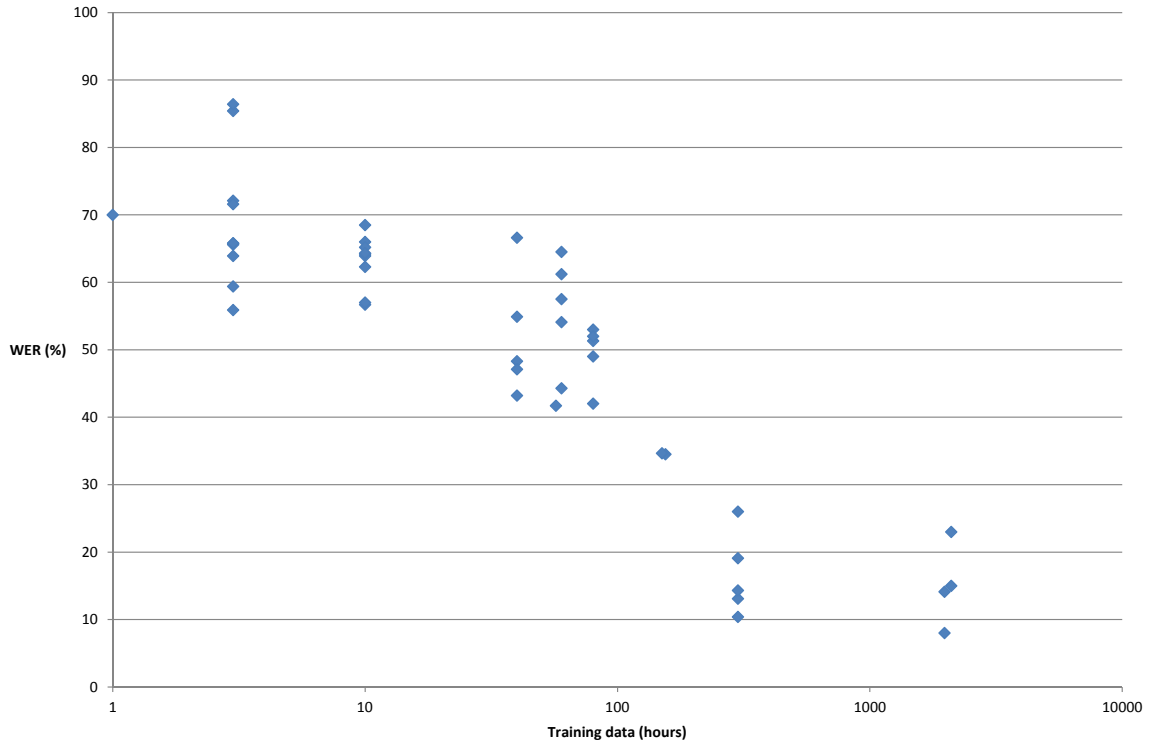

Figure 1-1: WERs of various conversational telephone transcription tasks. Numbers compiled from the following sources [19, 43, 46, 50, 53, 78, 82, 84, 86]

Lowering the amount of the training data not only directly impact the robustness of the acoustic model, it also indirectly limits the amount of vocabulary the system is exposed to, leading to a high Out-of-Vocabulary (OOV) rate, a rate which describes how often a spoken word is not included in the vocabulary of an ASR system. Since ASR systems can only generate a hypothesis that exists in its vocabulary, the OOV rate imposes a lower-bound on how low the WER can be. Moreover, the language models, which require large amounts of training text to estimate robust models, also degrade in a low resource setting. These combined issues lead to the undesirable performance of ASR systems when the training data is small.

1.2 Low Resource Languages

There is no single official definition on what makes a language low resource, and the meaning usually shifts depending on the target application. A language can be considered low resourced when typical methods can no longer be used, or they perform poorly. In general, a language with fewer native speakers tends to have less available data. Only around 400 languages out of the 7,000 languages have more than 1 million speakers [49]. However, languages with over millions of speakers can be considered low resource too in a time critical application such as responding to a natural disaster. With the time constraint, it is often hard to collect the sufficient amount of resources. For applications such as machine translation, which requires parallel text in both languages, many language pairs can be considered low resourced too even if the language themselves have many speakers [67].

From an ASR perspective, a low resource language can be considered as a language that is missing or lacking any of the three main components of a speech recognizer mentioned earlier. For example, the recognizer described in [78] uses around 2000 hours of transcribed speech, a lexicon containing more than 30,000 words, and a text corpus of 24 million words. Of the three resources, transcribed speech recordings are often considered the hardest and most time consuming part to collect, especially in languages with fewer speakers. Having fewer speakers not only implies the lower amount of speech recordings, but it also makes it harder to find people to transcribe those recordings. A pronunciation dictionary can sometimes be limited or not easily available in certain languages, since it requires an expert linguist to create. Finally, the text collection required for language modeling is often lacking when dealing with colloquial speech since it requires transcribed speech rather than web text.

1.3 Automatic Speech Recognition for Low Resource Languages

Researchers have been using multilingual resources to tackle the low resource problem. While a system trained solely on resources from a single language, e.g., a monolingual system, can perform poorly, a system trained on a pool of resources from various languages, e.g., a multilingual system[1], can provide substantial gains in performance [80]. Researchers often refer to the pool of languages, which can be either a low or rich resource, used in training as *source languages*, while the language of the target application is referred to as the *target language*. Approaches such as in [6, 69] try to learn a common lower-dimensional subspace across languages in order to reduce the amount of parameters that need to be learned for the target language. The term *cross-lingual* is often used in this type of methods for the case when a model is first trained in one language and then applied or adapted to another.

Recent progress on Deep Neural Networks (DNNs) has greatly improved ASR performance on many tasks [35]. One way to apply DNNs in ASR is via Bottleneck (BN) features [75, 92, 97, 98]. In this approach, a standard DNN with one smaller hidden layer, called the bottleneck layer, is trained. Then, the outputs of the bottleneck layer are used in conjunction with other features to train a standard recognizer that uses Gaussian Mixture Models.

Researchers have also used BN features to leverage out-of-domain resources, that are either multilingual [93, 95], or cross-lingual [85]. With access to larger amounts of data, the DNN can make use of the larger variability seen in training to extract better features, and ultimately improve the performance on the target language. These approaches not only alleviate the lack of training data, they also save the amount of time required to train DNNs for the target languages.

When multilingual resources are rich and diverse, one question that arises is how to best take advantage of the resources. Although it can be beneficial to use more

[1]Not to be confused with a system that can output multiple languages. In this thesis, our multilingual systems output one language just like monolingual systems.

of the available languages [26], there is also evidence that a source language that is close to the target language is more beneficial than a random one [29].

In this thesis, we propose methods to identify the best source languages for each target language, and to train better BN systems for low resource languages. These methods can also be used in systems that are not based on BN features such as hybrid-DNNs.

1.4 Main Contributions

The main contributions of this thesis can be summarized as follows:

- Developed techniques for building ASR systems that work on a variety of low-resource languages with minimal human intervention.
- Proposed a method for extracting better feature representations using DNNs. This technique achieved state-of-the-art results on monolingual tasks. It can also be easily applied in a multilingual framework to help improve ASR systems for low resource languages.
- Investigated the use of transfer learning on low resource languages.
- Investigated the need for selecting appropriate data for multilingual training. We proposed two methods that can select at the language level or the frame level to isolate acoustical and linguistical effects. Results showed significant improvement over naive multilingual systems on both transcription and keyword spotting (KWS) tasks.

1.5 Thesis Overview

The remainder of this thesis is organized as follows:

- Chapter 2 reviews concepts related to ASR, KWS, and DNNs. It also provides descriptions about the corpus and metrics used in the thesis.

- Chapter 3 goes over the details on how to build a robust monolingual system for both transcription and keyword spotting applications.
- Chapter 4 describes a basic multilingual system based on prior works which inspires the work in this thesis.
- Chapter 5 describes our proposed method to extract better feature representations and how it can be used in a multilingual setting.
- Chapter 6 investigates transfer learning for low resource languages and how data selection plays a significant role in performance.
- Chapter 7 summarizes the key concepts of the thesis and provides directions for future work.

Chapter 2

Background

2.1 Introduction

In this chapter we will describe some of the building blocks used in this thesis ranging from ASR basics to DNN usage in an ASR system. We will also describe the Babel corpus which we use heavily in this thesis.

2.2 Automatic Speech Recognition

Automatic Speech Recognition (ASR) is the process of transcribing speech automatically using machines, and is typically comprised of three main components, the lexicon, the acoustic model (AM), and the language model (LM).

The speech signal is typically transformed into a sequence of observation, X, that capture short-term temporal-spectral fluctuations. Typically A feature vector is extracted on a small time window of speech usually on the order of 25 ms. These windows, or frames in ASR jargon, are usually computed every 10 ms. Standard features used in ASR includes the Mel-frequency cepstral Coefficients (MFCCs) and Perceptual Linear Prediction (PLP) [12].

Given the observation sequence, X, extracted from a speech waveform, we want to find the best word sequence W^* that maximizes the posterior probability $\mathsf{P}(W \mid \mathsf{X})$

ass

$$W^* = \mathrm{argmax}_W P(W \mid X) \tag{2.1}$$

Using Bayes' Rule we can consider the above equation as

$$\begin{aligned} W^* &= \mathrm{argmax}_W \frac{P(X \mid W)P(W)}{P(X)} \\ &= \mathrm{argmax}_W P(X \mid W)P(W) \end{aligned} \tag{2.2}$$

where the term $P(X)$ can be simply dropped since it is fixed in the maximization. The first term $P(X \mid W)$ is generally referred to as the likelihood of the data.

Note that the likelihood term is usually modeled by Hidden Markov Models (HMMs) where the output probabilities are generated from Gaussian Mixture Models (GMMs). In this manner, a sequence of frames is generated from a sequence of hidden states, S. These hidden states typically represent a subword/phonetic segmentation of each word. For example, the word "cat" can be represented using a sequence of three hidden states each representing the phonemes, /k/, /æ/, and /t/, respectively. In other words, we would like to model

$$W^* = \mathrm{argmax}_W \sum_S P(X, S \mid W)P(W) \tag{2.3}$$

Under the Markov assumption, the equation becomes

$$W^* = \mathrm{argmax}_W \sum_S \prod_t P(x_t \mid s_t)P(S \mid W)P(W) \tag{2.4}$$

where x_t and s_t are the observation and hidden state at time t, respectively. These three terms are typically considered the main building blocks of a speech recognizer. $P(x_t \mid s_t)$ is called the Acoustic Model and provides a scoring mechanism for the frame observations given the state sequence. $P(S \mid W)$ is called the lexicon and provides a mapping between words and subwords/phonemes. $P(W)$ is typically modeled by a Language Model. In practice, the hidden states are typically modeled by 3-state triphones. A triphone, a phone with a left and right phonetic context, is used in

order to handle co-articulation effects. Each triphone is also usually modeled by 3 left-to-right states to handle the transient acoustic dynamics such as coarticulation during the production of a phone in context. Systems that model triphones (or any phoneme tuples) are usually referred to as having *context dependent* (CD) models, while systems that model just the single phonemes without any context use *context independent* (CI) models.

The parameters of the HMM acoustic model can be estimated in a Maximum Likelihood (ML) fashion using the Forward-Backward algorithm. At test time, the maximization can be solved by using the Viterbi Algorithm. Readers are invited to refer to [68] for more details.

2.2.1 Discriminative Training

In practice, the models trained using the ML criterion do not yield the best results, since the model conditional independence assumptions do not hold for speech data. To improve performance, discriminative training methods have been proposed. Instead of maximizing the likelihood of the data, discriminative training tries to minimize the confusion in the training data. In general, discriminative training relies on constructing an objective function that captures the degree of confusion and modifies the model parameters according to the objective. Commonly used objectives are Maximum Mutual Information (MMI) [1], Minimum Phone Error (MPE) [61], and state-level Minimum Bayes Risk (sMBR) [62], which try to minimize the errors in the frame, phone, and state level, respectively. We mainly use sMBR in this thesis, since it is often reported to yield the best performance [14, 94]

2.2.2 Lattice and N-best

In a practical speech recognizer, the number of hidden states can be on the order of tens of thousands. This makes solving for the best path in the HMM infeasible to be done exactly. ASR systems usually employ a time-synchronous beam search with pruning to help reduce the computation load. In the beam search, only a fixed

number of alternatives are considered at each time frame, while the paths with lower scores are discarded. This makes the beam search a trade-off between computation and correctness. The final product of the beam search is a lattice or graph, which represents a set of possible transitions of the hidden states with their associated scores. The best scoring path through the lattice is called the "best path" or the 1-best output. This is generally the main output of the speech recognizer. Depending on the application, the recognizer can also generate N-best hypothesis sentences generated from N best scoring paths through the lattice.

2.2.3 Language Models

The standard Language Models (LMs) used in ASR are n-grams. N-gram models represent the probability of generating the next word given the previous $N-1$ words. The LM usually assumes a fixed set of possible words. These words are considered in-vocabulary (IV). The number of words in the vocabulary is often referred to as the vocabulary size. Any word outside of the vocabulary is called an Out-Of-Vocabulary (OOV) word. Generally, any OOV word cannot be generated by the ASR.

Recently, researchers have looked into using Recurrent Neural Networks (RNNs) for language modeling as well [57]. However, due to the recurrent nature of the model, LMs based on RNNs are only for re-scoring the lattice or the N-best list after the beam search is completed.

One metric to evaluate language models is to compute the perplexity on heldout data, defined as:

$$\text{Perplexity} = 2^{-\frac{1}{n}\log_2(P(W))} \tag{2.5}$$

where n is the number of words in the heldout data, and $P(W)$ is the probability of the data estimated by the model [73].

2.2.4 Word Error Rate (WER)

WER is a standard metric used to measure ASR performance. It can be considered as the word-level edit distance from the ground truth transcription. It is composed of three kinds of errors, substitution errors, insertion errors, and deletion errors.

$$\begin{aligned} \text{Substitution Error} &= \frac{\#\text{ of substitution errors}}{\#\text{of ground truth words}} \\ \text{Insertion Error} &= \frac{\#\text{ of insertion errors}}{\#\text{of ground truth words}} \\ \text{Deletion Error} &= \frac{\#\text{ of deletion errors}}{\#\text{of ground truth words}} \\ \text{WER} &= \text{Substitution Error} + \text{Insertion Error} + \text{Deletion Error} \end{aligned} \tag{2.6}$$

WER is often reported as a percentage. A perfect transcription will have 0% WER. Note that since we can have insertions, WER can be higher than 100%.

For Cantonese and other character-based languages, WER is often calculated at the character level instead of the word level (Character Error Rate or CER), but for the sake of simplicity we will use the term WER for Cantonese as well.

One caveat of the WER metric is that it weights all errors equally. A substitution of "Sheet" with "Cheat," which sound very similar, counts the same as substituting "Sheet" with "Kittens." In this sense, under certain conditions such as high WER, an improvement in WER does not necessarily mean a better system.

2.3 Acoustic Features and Feature Transforms

In this section we describe some of the features used in this thesis. We will also discuss some standard techniques that can be used to improve those features.

2.3.1 Perceptual Linear Prediction features

One of the features we used in most of our systems is Perceptual Linear Prediction (PLP) coefficients. The Perceptual Linear Prediction (PLP) method was proposed by Hermansky in [33]. It is a based on Linear Predictive Coding (LPC), which tries

to model the spectral envelope of a signal using an all-pole model [54]. The main difference is that PLP tries to model a *transformed* spectrum that tries to mimic the human auditory system. The extraction process of the PLP features is summarized in Figure 2-1.

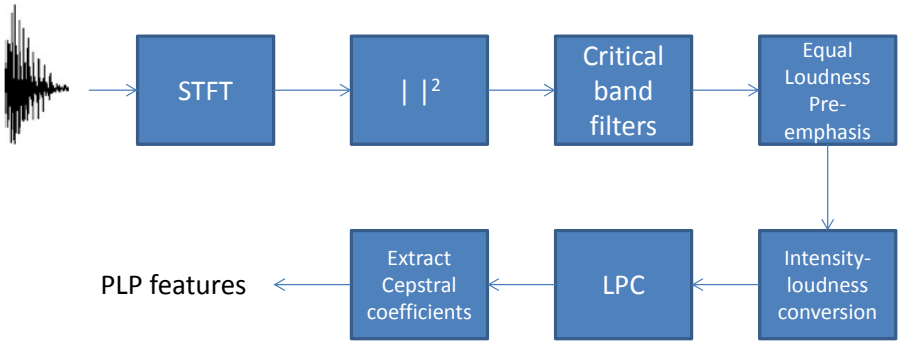

Figure 2-1: PLP features extraction pipeline. The input waveforms are windowed and passed through a STFT for time-frequency analysis. The critical band filters, equal loudness pre-emphesis, and intensity loudness conversion are used to transform the spectrum in order to mimic the human auditory system. LPC can then be used to approximate the transformed spectrum. Finally, cepstral coefficients can be computed for each input window.

First, Short-Time Fourier Transform (STFT) is computed on each frame to extract the information in each frequency bin of the signal. We take the power of the complex-valued output to estimate the power spectrum. The power spectrum is then passed through Mel-frequency filterbanks. The Mel-frequency is an estimate of the frequency scale that is perceived by humans [59]. An example of the Mel-frequency filterbanks is shown in Figure 2-2. Then, an equal loudness pre-emphasis is applied to amplify the power of the higher frequencies. The equalized values are transformed according to the intensity-loudness power law by taking the cubic root [15]. LPC is applied on the normalized power spectrum values. Finally, cepstral coefficients can be extracted from the LPC values by solving a set of recursive equation [37]. The benefit of being in the cepstral domain, the log of the power spectrum, is that one can apply mean subtraction to remove channel effects.

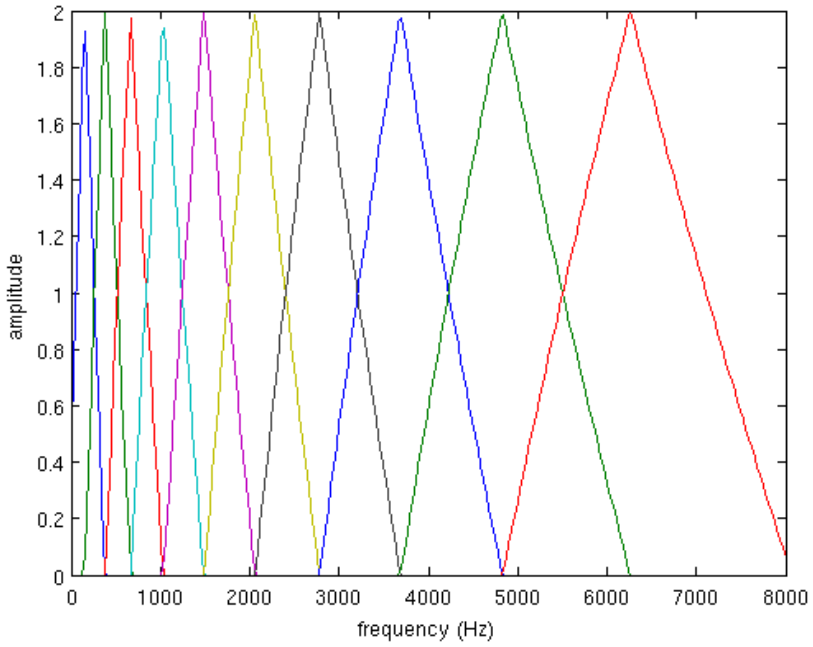

Figure 2-2: Coefficient values of Mel-frequency filterbanks. The filters are wider in the higher frequencies to imitate human perception.

2.3.2 Cepstral Mean and Variance Normalization (CMVN)

CMVN is a method use to normalize the effects due the difference in the recording conditions, such as different microphones or recording rooms. These recording effects are often model as a convolutive noise in the time domain. The convolutive noise becomes an additive noise in the cepstral domain. One method to normalize the convolutive noise is to remove the means in the cepstral domain. This can be done by removing the mean, accumulated over the entirety of each recording, in each cepstral coefficient. Researchers also often normalize the features to unit variance so that they fit better to the GMM models.

2.3.3 Vocal Tract Length Normalization (VTLN)

Due to the difference in size of the vocal tracts between each person, the speech produced by each talker are slightly different. Differences in vocal tract length are related to resonance frequencies, e.g. formant frequencies. To account for the changes,

(VTLN) was proposed by Cohen *et al.* to normalize the effect [11]. VTLN can be implemented by scaling the frequency axis in the filterbank analysis mentioned previously to normalize the difference shifts introduced by each speaker's vocal tract. To estimate the scaling factor for each speaker, one can search through different scaling factors and compare likelihoods obtained from the GMM-HMM model.

2.3.4 Delta Features

One of the key assumptions of the HMM model is conditional independence: given the hidden state, observations are independent of each other. However, this assumption does not hold for ASR where adjacent frames can be highly correlated. Moreover, some phonemes are characterized by their dynamics rather than a single static snapshot that is captured by a single frame. To alleviate this, researchers often additionally compute derivatives or "delta" features (Δ), which capture the dynamics of the original features. Delta features are often calculated by computing the difference between features of the previous and the next frame. Higher order derivatives, such as the delta-delta (Δ^2), can also be computed from the delta features and used in the same manner. Delta and delta-delta features are typically concatenated with the original features to create a large feature vector.

2.3.5 Dimensionality Reduction and Decorrelation Techniques

ASR models often use diagonal covariance GMMs. However, the features used in speech recognition can be highly correlated. Sometimes the features extracted can also have too many dimensions to model sufficiently with existing data. Researchers often use linear transforms such as Principle Component Analysis (PCA) or Linear Discriminant Analysis (LDA) to perform dimensionality reduction and whiten the data [4]. PCA, which is an unsupervised method, tries to find directions of maximum variability in the data, is mainly used for dimensionality reduction of single frames. On the other hand, LDA, which attempts to find directions that can best separate the classes, are often used to perform dimensionality reduction on a set of features

concatenated from multiple frames, i.e. stacked frames. The class labels used to perform LDA estimation are the hidden state labels.

2.3.6 Maximum Likelihood Linear Transform (MLLT) and Feature-Space Maximum Likelihood Linear Regression (fMLLR)

Another linear transform that is often used in ASR is MLLT. The objective of MLLT is to find a global linear feature transform that maximizes the average frame level likelihood given the model, i.e. the HMM-GMM. MLLT, LDA, and the HMM-GMM parameters are often estimated together iteratively. Concretely, first, train a HMM-GMM model, then estimate the LDA and MLLT transforms. At certain iterations of the HMM-GMM training, re-estimate the LDA and/or MLLT transforms.

fMLLR is an *affine* transform on the input features that also tries to maximize the likelihood of the model. fMLLR is typically used to estimate speaker specific transforms for speaker adaptation, while MLLT is applied on globally on every speaker. In fact, MLLT estimation can be considered as a simplified version of fMLLR (if estimated on the whole dataset). Many of the experiments in this thesis deals with long recordings on the order of 5 minutes, making fMLLR a very powerful tool. fMLLR is also known as Global Constrained Maximum Likelihood Linear Regression (CMLLR) in the literature [63]. For more details on how these feature transforms can be estimated and used in practice refer to [21].

2.4 Keyword Spotting (KWS)

One of the downstream tasks that uses the output generated from an ASR system is Keyword Spotting (KWS). Keyword spotting is a detection task that tries to search for all occurrences of the key term or phrase in a spoken database with the lowest number of false alarms. KWS can be used for search and retrieval of terms in a large database. KWS can also provide acceptable performance even when the WER is as

high as 70%, making it an ideal application for low-resource language ASR. A human user can search for a particular key term and listen to the snippet of the audio rather than go through the entire database.

In this thesis, a keyword can be just a simple word, or a whole phrase. The simplest solution for KWS is to search for the 1-best output from the ASR system by string matching. This usually yields undesirable results since the ASR system is erroneous. A smarter solution is to search in the lattice or N-best which is more likely to contain the correct transcript. By searching in the lattice, the KWS performance can be very accurate even when the 1-best results get every other word incorrect. The score of a keyword hit is typically the score or probability associated with the lattice arc of the keyword. The score is then compared against a threshold to decide whether or not the hypothesis should be accepted. An accepted hypothesis is often called a keyword *hit*.

2.4.1 Averaged Term Weighted Value (ATWV)

ATWV is a standard metric to evaluate KWS tasks [18]. ATWV views KWS as a detection task, where the score is calculated based on the trade off between the percentage of correct detections and the number of false alarms.

The equation to calculate the ATWV is as follows:

$$ATWV = 1 - \frac{1}{K}\sum_{k=1}^{K}\left(\frac{\#miss(k)}{\#ref(k)} + \beta\frac{\#fa(k)}{T - \#ref(k)}\right) \tag{2.7}$$

where K is the total number of keywords, $\#miss(k)$ is the number of reference tokens for keyword k that are not detected, $\#ref(k)$ is the total number of reference tokens for keyword k, $\#fa(k)$ is the total number of false alarms for keyword k, T is the total length of the audio in seconds, β is the trade-off between false alarms and miss detections which is set to 999.9.

A perfect ATWV is 1.0, while a system that yields no output will have 0.0 ATWV. A system with too many false alarms can have a negative ATWV.

A related metric to the ATWV is the *Maximum* Term Weighted Value (MTWV),

which is defined as the maximum ATWV over all decision thresholds. In this thesis, we will mostly use MTWV as a metric when evaluating acoustic models. In practice, ATWV and MTWV are usually very close, since estimating the correct threshold can be done accurately using the development set.

As mentioned earlier, KWS is usually done on the lattices instead of the 1-best results like in the transcription task. Thus, for higher WER tasks ATWV is considered to give a better metric for the system in question.

There are two caveats regarding how ATWV is calculated. First, rare terms are weighted more per occurrence. Missing one rare term can hurt ATWV more than missing one term that appears thousands of times. This makes sense from the user stand point. This also emphasizes the weak points of statistical models where rare terms are harder to model properly. Many methods are needed to circumvent this problem, from generating larger lattices so that rare terms appear in the lattice instead of getting pruned away, or using a different pruning threshold for rare words.

Second, the threshold used in the KWS is a global threshold. However, longer or rarer terms usually have lower scores than shorter ones. Thus, it is also important to apply normalization techniques so that the scores are comparable.

2.4.2 Score Normalization

To compensate for the discrepancy in scores for each keyword, it is important to normalize the score of each hit. Let $h_{i,k}$ be the score of the i^{th} hit for the k^{th} keyword, we compute the normalized score $h'_{i,k}$ using the following methods:

- Sum-to-one normalization [55]

$$h'_{i,k} = \frac{h_{i,k}}{\sum_i h_{i,k}} \tag{2.8}$$

- Exponential Normalization with Keyword-specific Thresholding [42]

$$thr(k) = \frac{N_{true}(k)}{T/\beta + \frac{\beta-1}{\beta} N_{true}(k)} \tag{2.9}$$

$$h'_{i,k} = h_{i,k}^{(-\frac{1}{\log(thr(k))})} \tag{2.10}$$

where $N_{true}(k)$ is an estimate of how many time the keyword k appears in the transcript. $thr(k)$ is the estimated keyword dependent threshold which can be used to scale $h_{i,k}$. N_{true} can be estimated by summing all the posteriors of the hits for keyword k scaled by some factor to account for occurrences that were missed by the recognizer.

In practice, these methods do not differ much in terms of performance as long as you use one of them. We use the Exponential Normalization with Keyword-specific Thresholding for the rest of this thesis.

2.5 Deep Neural Networks (DNN)

Recently there has been an explosion of interest in Neural Network models for AM, LM, and the entire ASR pipeline. DNNs, just like the name suggests, are a variant of Neural Networks, where there are many hidden layers. This was made possible by the recent availability of more powerful hardware such as GPUs. glsDNN computations typically require many large matrix multiplications which can be parallelized easily in the GPU. In ASR, researchers have shown 5-30% relative improvement on various tasks [35]. The power of DNNs are often believed to come from the fact that the representation and the classification are trained jointly. In the following segments we will discuss the components of the neural network and how it can be trained and used in an ASR setting.

2.5.1 Neurons

Neurons are the most basic unit of a DNN. A neuron is usually characterized by a set of input weights W, a bias b, and a non-linearity function F. In general, a neuron

computes

$$o = F(WX + b) \tag{2.11}$$

where o is the output of the neuron. An example neuron is shown in Figure 2-3.

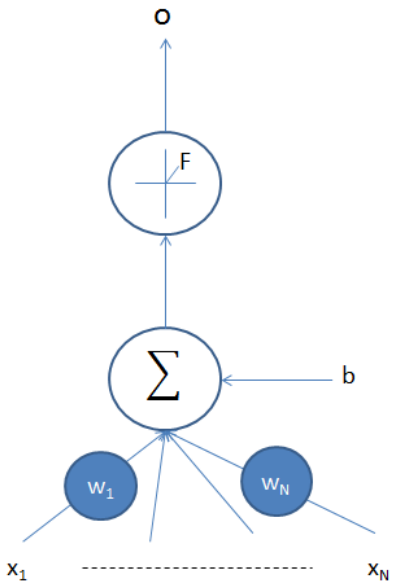

Figure 2-3: An illustration of a neuron. First, a weighted combination of the input, X, with a bias term, b, is computed. It is then passed through a non-linearity function, F, yielding the output, O. A hidden layer is formed by multiple neurons. A DNN is a series of hidden layers, where the output of the previous layer is the input of the next layer.

A neural network layer consists of a number of neurons taking the same set of input X. A deep neural network can then be constructed by stacking multiple layers (typically more than 3 to be considered deep).

The non-linearity is important to ensure that the stacking of the hidden layers is not just a cascade of linear transforms. Many kinds of non-linearities have been used by the community. The Sigmoid non-linearity is the most popular within the ASR community. Rectified Linear Units (ReLUs) are also widely used since they offer a faster and more stable convergence. However, Sigmoids typically outperform ReLUs

slightly when the model converges.

$$F_{sigmoid}(x) = \frac{1}{1+e^{(-x)}}$$
$$F_{ReLU}(x) = max(0, x) \tag{2.12}$$

2.5.2 Softmax Layer

The final layer of the DNN is usually a classification layer called the softmax layer. For a N-class classification task, the softmax layer consists of N output nodes, y_i.

The classification output can be computed using the equation

$$y_i = \frac{exp(w_i x)}{\sum_{k=1}^{N} exp(w_k x)} \tag{2.13}$$

where w_i is the weight vector associated with the i^{th} output.

2.5.3 Neural Network Training

A neural network can be trained using an algorithm called Stochastic Gradient Descent (SGD) [35] (Other variants and methods also exist, but SGD is most widely used in practice). SGD is a gradient descent approach with a slight modification where the true gradient is estimated by the gradient of a small subset of the training examples, called a mini-batch. When the learning rate is controlled appropriately SGD is guaranteed to converge to a local minima. The algorithm for SGD can be summarized as follows:

Algorithm 1 Stochastic Gradient Descent Algorithm

1: Initialize the weights and bias, θ, of neural network
2: **repeat**
3: Shuffle the training set, partitioning it into M mini-batches
4: **for all** mini-batch, m, **do**
5: Compute the objective given the current weights $O(\theta, m)$
6: Compute the gradient from the objective $\nabla O(\theta, m)$
7: Update the model according to the learning rate n
$w := w - n\nabla O(\theta, m)$
8: **end for**
9: **until** some criterion is achieved.

Each pass through the entire training data, i.e., step 3, is called an epoch. The stopping criterion is usually the objective function computed on some heldout set. Note that it is important for speech applications that the input examples are shuffled to get a good estimate of the objective function. Adjacent input frames are highly correlated, i.e. same phonemes. Frames from the recordings are also correlated in terms of speaker, noise, and channel characteristics. Not shuffling the frames properly can lead to a degradation in performance, for example Su *et al.*. reported a 7% relative loss in WER for shuffling only utterances but not frames [87].

2.5.4 Cross Entropy Criterion

The objective function typically used in ASR is the Cross Entropy (CE) criterion. Assuming hard input label decisions (i.e. the probability of the target label taking a value of only 0 or 1), this can be calculated from the output of the softmax layer, y_{it}, and the target label d_t for each given frame t as follow

$$O = -\sum_{t} \log(y_{d_t t}) \tag{2.14}$$

2.5.5 Neural Network Initialization

There are many ways to initialize the DNN. Since SGD is only guaranteed to find the local minima, initialization can play a crucial role. Researchers employ various methods to initialize the network, such as unsupervised pre-training [36]. For larger speech tasks (more than 100 hours of training data), pre-training usually does not provide additional gains. A simple initialization heuristic is typically enough. In this work, we follow the initialization method by [24] where the weights and biases are initialized according to the number of input and output connections of the hidden units (fan-in and fan-out) from the Uniform distribution according to the following equation:

$$W \sim \mathtt{Uniform}(0, \frac{1}{\sqrt{\mathtt{FanIn} + \mathtt{FanOut}}}) \tag{2.15}$$

2.5.6 DNN Training in Practice

In practice, many researchers apply tricks to increase the training speed and avoid bad local minimas.

- New Bob algorithm [16]

  The New Bob algorithm is a training schedule where the training is done until the objective on the heldout set is worse than the previous epoch. When this happens, the learning rate is halved and the training restarts with the model in the previous epoch. This method typically improves the performance on TIMIT by 5-10% relative.

- Momentum

  To speed up training we also apply momentum, where part of the gradient from the previous mini-batch is kept for the next update. The momentum is used after the first epoch. The momentum is also reset at every halving step.

- Class balancing (silence removal)

  Silence can total to more than 30% of the training frames. This large imbal-

ance of the labels can degrade the performance of the DNN. In practice, we find keeping just 5 frames of silence before and after speech frames improves performance and reduces the training time.

2.5.7 DNN in Acoustic Modeling

Putting it all together, we can now train a DNN given input frames and output labels. In ASR, the output labels are typically the hidden state labels, as described in Section 2.2. Thus, training of a DNN-based ASR system usually starts with a baseline speech recognizer for frame-level output target labels. This is usually done by force aligning the transcription with the input speech. The input frames are typically MFCCs, filter banks outputs, or Bottleneck features [35]. This means that the DNN is trying to emulate the Acoustic Model $P(x_t \mid s_t)$ in Equation 2.4. However, since DNNs typically produce posteriors rather than likelihoods, we need to normalize the DNN outputs using the class priors. This method of using the DNN for acoustic modeling in ASR is usually called the "hybrid DNN-HMM."

Another approach widely used is called the Tandem approach [27]. Similar to the hybrid DNN-HMM approach, the DNN is trained using the hidden state target labels. However, instead of using the target output of DNNs to replace the likelihoods, the DNN is used to generate input features to feed into another generic HMM speech recognizer (GMM-HMM). The input feature can be extracted from the outputs of some hidden layer in the DNN. Typically that layer is forced to have a smaller number of neurons, i.e. a bottleneck (BN), to force the DNN to encapsulate all the relevant information into a smaller dimensionality that is preferred by a typical GMM-HMM system. The extracted features are concatenated with regular ASR inputs such as MFCCs and used *in Tandem* for the new GMM-HMM. Since the DNN trained the input representation in a discriminative manner, this usually outperforms handcrafted features such as MFCCs and PLPs. The hybrid DNN-HMM and the Tandem approach are depicted in Figure 2-4.

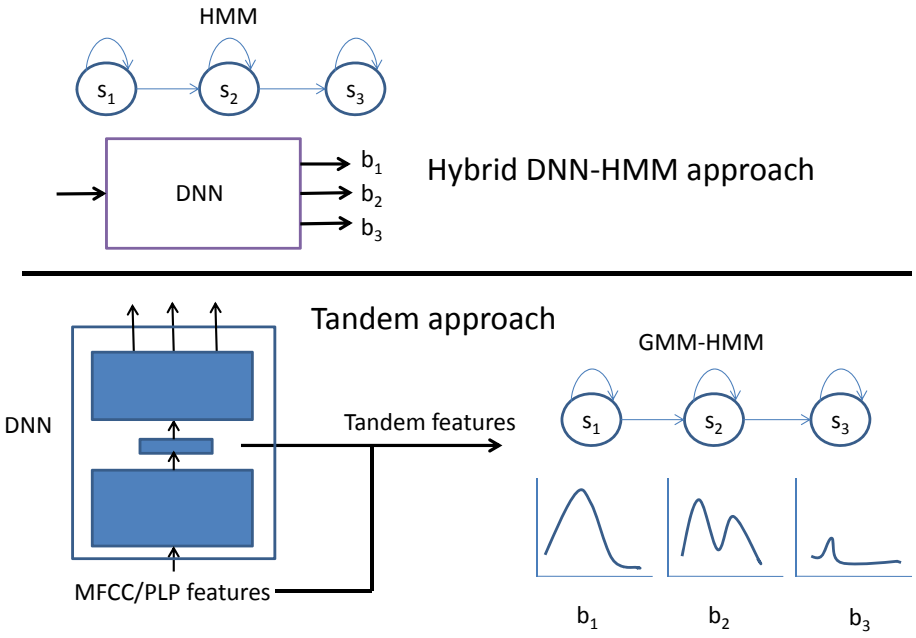

Figure 2-4: Two approaches for using DNNs in ASR, the Tandem and hybrid approach. The top part of the figure shows the hybrid DNN-HMM approach where the observation probabilities are generated by the DNN. The bottom shows the Tandem approach. A DNN with a bottleneck layer is used to extract BN values which are combined with traditional features to use as input to a traditional GMM-HMM.

2.5.8 Hybrid vs. Tandem

The pros and cons of the two approaches are summarized in Table 2.1.

Table 2.1: Comparison between Hybrid and Tandem approaches.

Hybrid	Tandem
• Less training steps • Typically performs better in CE • Cannot use existing HMM-GMM techniques • Output is task specific	• More training steps • Performs worse in CE due to the BN • Can build on existing techniques such as speaker adaptation • Output (the BN) can be used in other tasks

In this thesis we will mostly use the Tandem approach. However, many techniques can also be applied in the Hybrid approach, and ultimately the combination of the two approaches work the best, as we will report later.

2.6 IARPA-Babel Corpus

Most of the experiments in this thesis use the Babel corpus. The Babel program is a project funded by IARPA with a goal to improve ASR technologies for low resource languages. The program has two main focii. First is to improve the ASR and KWS spotting capability with limited human supervision and supervised resources such as transcribed speech, lexicon, domain knowledge, etc. Second, it aims to reduce the time to develop such a system in the face of some unforeseen time critical crisis, since typical ASR systems typically require development time to identify and cope with special language characteristics in order to reach a certain performance threshold. To this end, the program emphasizes the need for technology that can be applied to every kind of languages in a fast and efficient manner.

Currently the corpus consists of 24 languages from all over the world: Amharic, Assamese, Bengali, Cantonese, Cebuano, Dholou, Guarani, Haitian, Igbo, Javanese, Kazakh, Kurmanji, Lao, Lithuanian, Mongolian, Pashto, Swahili, Tagalog, Tamil, Telugu, Tok Pisin, Turkish, Vietnamese, and Zulu. As shown in Figure 2-5, the

languages in the Babel corpus span many regions around the globe, covering various language families and writing systems.

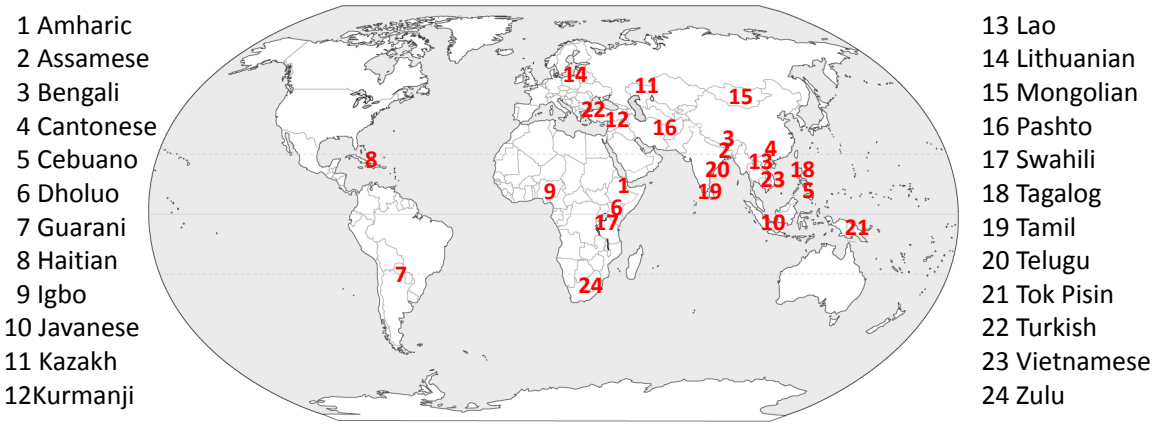

Figure 2-5: Languages available in the Babel corpus marked by location of recording.

2.6.1 Corpus Structure

The corpus structure for each language in the Babel corpus is organized in a similar manner. The recordings consist of two main types, prompted speech, and conversational speech. The prompted recordings are short answers to various kinds of questions such as "What is the current time?" "Read this address" "What is your name?" The second kind of recordings is conversational speech. In this type, the volunteer calls someone he or she knows and start a conversation. This type of recording is usually 10 minutes per conversation (2-sided for 20 minutes in total). The volunteers are asked to make this call at their own choosing meaning they can be in a car, restaurant, using landlines or telephones, and the other side of the call can be in a different kind of environment. Since these conversations are recorded over telephone lines, they are sampled at 8kHz.

Some conversational recordings are collected in a specified manner, e.g. with a predetermined microphone, in a specific room and noise condition. These recordings are recorded 48kHz. From the nature of the sampling rate, we will call this type of data the wideband recordings, and the 8k recordings, narrowband recordings. There are up to 8 pre-determined microphone types in each language (some unseen in training

and dev). These wideband recordings are usually less than 10% of the whole corpus and pose a very difficult challenge. In this thesis, we treat the wideband recordings the same way we treat the narrowband recordings. The wideband recordings are downsampled to 8kHz.

Each language's recordings also contains various amounts of transcribed data. The scripted speech are all transcribed. However, only a portion of the conversational data is transcribed. This ranges from 40 to 80 hours depending on the language (including silence, the actual speech amount is lower). The rest of the recordings are un-transcribed. The un-transcribed data can range from 0 to 40 hours.

Every recording is marked with speaker gender, dialect, recording device, recoding environment, and telephone provider. The transcriptions also mark hesitations (ah, umm), noise, laughter, cough, and other non-speech artifacts such as background noise or dial tones. This makes the Babel corpus very flexible for various kinds of speech related techniques and tasks.

Each language consists of 3 standard subsets called training, (train) development (dev), and evaluation (eval). The dev and eval sets only contain conversational recordings.

2.6.2 Lexicon

The lexicon for every language is provided in SAMPA phonetic format. SAMPA is a phonetic symbol set that is entirely ASCII compatible, unlike IPA. Some examples of the lexicon in the corpus can be found in Figure 2-6.

Pashto:

اخراجات AixorAjAto e x . 4 A . " dZ A t

Cantonese:

一三二零 yat1saam1yi6leng4 j 6 t _1 . s a: m _1 . j i: _6 . l E: N _4

Figure 2-6: Example lexicon entries in the Babel corpus.

The first column is the original script. The second column is the Romanized

version of the original script (if the language uses non-Roman characters). Then, it is followed by the SAMPA phones. Syllables are marked with . And ‘ marks stressed syllables. Tones are indicated by numbers preceded by _. If multiple pronunciations exist, all will be provided (to the extent possible). Note that SAMPA phones are not standardized across languages in the Babel corpus, meaning the same symbol might represent different phonemes or tones across languages. However, most are relatively consistent.

2.6.3 Evaluation Metric

The target applications for the Babel program are automatic transcription and keyword spotting (KWS). For the two tasks, the program uses two standard evaluation metrics, the WER and ATWV mentioned in Section 2.2.4 and 2.4.1.

2.6.4 Evaluation Keywords

There are two main lists of keywords for each language, the dev keywords (kwlist2) and the eval keywords (kwlist4 or kwlist5). The dev keywords are automatically generated from the training set transcription by the method mentioned in [90]. The eval keywords are provided by the Babel program for the purpose of the evaluation. There are around 2000 dev keywords for each language, while the eval keywords range from 3000 to 7000 keywords. Each keyword can be a single word or a phrase, since some of these are automatically generated, they are not guaranteed to be meaningful keywords.

2.6.5 Languages and Training Packs

Table 2.2 summarizes the languages and the data amount for each language.

The Babel corpus defines 4 different training conditions, each with varying amount of training data:

Table 2.2: Language statistics in the Babel corpus for the FLP condition. Tones are the amounted of tones indicated in the lexicon provided. Zulu is tonal but not marked in the Babel corpus. The hours column is the total sum of segments of audio that contains speech based on the transcriptions. Segments in Cantonese include large amounts of silence. The total amount of real speech is around 72 hours. "Wideband" indicates whether the language pack contains some amount of wideband recordings. "Graphs" indicates the number of unique characters in the training pack. The "Words" column shows the number of words in the transcription, while "Vocab" only counts unique words.

Language	Phones	Tones	hours	Speakers	Wideband	Graphs	Words	Vocab
Cantonese	37	6	141	952	no	3322	892k	19939
Vietnamese	68	6	88	954	no	100	923k	5439
Tagalog	48	n/a	85	966	no	35	595k	22203
Pashto	44	n/a	78	959	no	54	888k	18750
Turkish	42	n/a	79	990	no	42	589k	40801
Bengali	53	n/a	62	720	no	92	481k	26531
Assamese	50	n/a	61	720	no	92	451k	23927
Zulu	47	n/a	62	718	yes	53	406k	60254
Haitian	32	n/a	67	724	yes	37	625k	13719
Tamil	34	n/a	69	724	yes	75	486k	58493
Lao	43	6	66	733	yes	59	597k	6614
Cebuano	28	n/a	41	478	yes	34	328k	15317
Kazakh	61	n/a	39	494	yes	71	270k	22291
Kurmanji	37	n/a	41	502	yes	38	346k	14317
Telugu	50	n/a	41	482	yes	68	267k	37646
Lithuanian	89	n/a	42	480	yes	43	351k	32586
Tok Pisin	37	n/a	39	480	yes	34	324k	6237
Swahili	38	n/a	44	491	yes	35	287k	25115
Mongolian	53	n/a	46	492	yes	42	403k	21016
Amharic	59	n/a	43	478	yes	253	281k	32815
Javanese	39	n/a	45	480	yes	33	309k	13860
Dholou	43	n/a	41	486	yes	34	360k	16488
Guarani	49	n/a	42	486	yes	49	311k	25270
Igbo	78	n/a	43	479	yes	34	490k	15872

Full Language Pack (FLP)

The FLP condition includes all the transcribed data listed in Table 2.2 and and the 10 hour dev sets used for both tuning and evaluation. There is also an official evaluation set. However, most of the transcription for the eval set are not available. Therefore, most research reports performance on the dev as if it is an evaluation set.

Limited Language Pack (LLP)

The LLP includes a 10 hour subset of the FLP. In selecting the subset of the data, whole recordings are selected and used in their entirely. The dev and eval set are the same as the FLP condition.

Very Limited Language Pack (VLLP)

This condition includes a 3-hour subset of the LLP. This condition also uses its own 3 hour dev set called the tune set. The construction of this subset is slightly different from the LLP. First, the recordings from the 10 hour LLP subset are divided for VLLP tune and VLLP train. Then, select equally from each recording (starting from the middle of the conversation) until the 3 hours is reached. This is done to ensure speaker diversity. The VLLP tune is constructed in the same manner. For the VLLP condition, the FLP dev can only be used for evaluation. Due to the selection process, VLLP tune is harder than the dev set. All VLLP training data are conversational.

Active Learning Language Pack (ALP)

This condition is designed for research in active learning, e.g. selecting the subset of data to transcribe that best improves the recognizer. The ALP starts with a fixed initial 1 hour of transcribed data (a strict subset of the training set in VLLP). Then, 2 hours of additional speech data can be chosen from the FLP training set (excluding the VLLP dev) to be transcribed and used in training for the final system. The dev set for the ALP is the same as VLLP dev. In this thesis, we will not work on this condition. Researchers found that using active learning, the ALP condition can

slightly outperform the VLLP [19]. However, the use of active learning in practice is still under debate, since it takes longer to transcribe snippets of sentences out of context than transcribing the full conversation. The gains from active learning also come mostly from vocabulary expansion, e.g. finding regions that contain OOV words. However, the gain diminishes with web data augmentation, which alleviates the OOV problem.

Note that the Babel corpus is an actively growing corpus. Languages and training packs have been regularly added over the 5 year period from 2010 to 2016. The choice of the language and training packs used in each experiment are often affected by the availability of the training packs at the time of the experiment. We present the work out of chronological order for a more coherent story, so there might be inconsistencies between experiments in terms of the training packs used. The exact version numbers for each training pack used can be found in Appendix B.

Chapter 3

Monolingual Systems

3.1 Introduction

In this chapter we investigate several issues related to ASR for low resource languages. Concretely, we will try to answer the following issues

- What would be ideal features that work for all languages?
- What is the extent of needing an expert lexicon?
- How can we cope with OOV terms?
- What gains do additional web data provide?

In answering these issues, we will also develop a set of techniques that will be used with our multilingual systems.

3.2 Multilingual Features

Features like MFCCs and PLPs have been widely used in the speech recognition community. These features, which were developed for English ASR systems, do not capture some information used in other language such as tones. We also would like features that are more robust to noise in real world environments. To this end we will investigate features that generalize well across different kind of languages.

3.2.1 Fundamental Frequency Features

Tones are associated with dynamics of the fundamental frequency (F_0), thereby making F_0 features useful for tonal languages. There are many methods to extract F_0. One common method is via autocorrelation, which computes the autocorrelation of the signal within a frame. The second highest peak of the autocorrelation will typically represent the fundamental frequency of the speech signal. Other more sophisticated techniques build on this observation for better accuracy such as tracking the peak of the autocorrelation across frames [88], or normalizing the autocorrelation according to the analysis window [5] so that it is more robust to noise.

F_0 feature is often coupled with the Probability of Voicing (PoV), which tries to predict whether the frame is voiced or unvoiced. PoV is usually a by-product of the autocorrelation, by using the value of the 2nd highest peak. This measures how strong the periodicity of the signal is. F_0 values and PoV are usually smoothed over time so that they fit better with the GMM-HMM framework. For this work we follow the implementation in Kaldi [22] to extract F_0 and PoV features.

3.2.2 F_0 Experiments

To evaluate the effectiveness of pitch features, we conducted experiments using the FLP of four languages Turkish, Tagalog, Vietnamese, and Cantonese, which were the languages available at the the time of the experiment. Vietnamese and Cantonese are tonal languages. We trained a 3-state CD HMM-GMM recognizer using Kaldi, an opensource speech recognizer [65]. We used 13-dimensional PLP features with delta and delta-delta coefficients. There are several fundamental frequency estimators in Kaldi. We selected "KaldiPitch" which has been reported to provide the best results. The features were normalized using Cepstral Mean and Variance Normalization (CMVN). For lexicons, we used the ones provided in the Babel corpus. Trigram language models were trained on the transcriptions from the training set. The standard tri5 recipe in Kaldi was used. This recipe includes Linear Discriminant Analysis (LDA) feature transform, speaker adaptation (fMLLR), and discriminative training

(sMBR) which we described in Chapter 2.

For Cantonese, we also trained two different versions. One used the lexicon provided in the corpus which has tone information, the other used a modified lexicon which no longer has tone information. Table 3.1 summarizes the results:

Table 3.1: WER comparisons between systems trained with and without F_0 and PoV features. Vietnamese and Cantonese are tonal languages, while Turkish and Tagalog are non-tonal. "Cantonese (no tone)" indicates a Cantonese system trained using a lexicon without any tone information.

Language	Without F_0 and PoV features	With F_0 and PoV features
Vietnamese	64.0%	58.0%
Cantonese	59.0%	53.5%
Cantonese (no tone)	59.3%	54.9%
Turkish	57.3%	55.0%
Tagalog	56.1%	54.2%

As shown from the Table, using F_0 features helps improve the WER on tonal languages by around 6%. They also help, to a lesser extent, non-tonal languages such as Turkish and Tagalog as well, making them a good feature to use in a multilingual ASR framework. Interestingly, losing the tone information in the lexicon only degrades the system slightly when no F_0 features are used. However, The gaps becomes 1.4% when F_0 are used. This shows that the F_0 features are providing the needed information to distinguish tones. From this point onwards, we will always include F_0 and PoV features in feature set.

3.3 Lexicon

One of the main problems with ASR for a low resource language is the need for a pronunciation lexicon. The lexicon is usually handcrafted by linguists who are experts on the language. It is also prone to errors, or omission of alternative pronunciations, which is especially challenging in languages with diverse dialects. In a time and resource limited scenario, crafting a lexicon from scratch is impractical. One possible alternative to the lexicon is to use graphemes as the phonetic representation. For

example, a graphemic lexicon entry for the word "apple" would be

apple: a p p l e

This makes lexicon creation a trivial task.

We conducted experiments to compare recognizers trained with and without the expert lexicon. The recognizer setup follows the previous section, except for the training data size, which is the VLLP condition (3 hours) to observe the effect of the graphemic lexicon in the extreme case. We randomly picked four languages which have the VLLP condition defined. We compare the graphemic lexicon with the expert lexicon on various languages in Table 3.2.

Table 3.2: WER comparisons between systems trained with a phonetic (expert) lexicon and a graphemic lexicon.

Language	Phonetic	Graphemic
Kazakh	76.8%	77.0%
Kurmanji	85.5%	85.1%
Telugu	86.3%	87.0%
Cebuano	75.7%	75.9%

As shown, the WER performance of the system trained with a graphemic lexicon is in line with the expert lexicon, and in one case the expert lexicon performs slightly worse. This trend is also observed in other training conditions such as the the LLP which has 10 hours of training data [47]. Note that many of these languages have a rather simple character to phoneme mapping. However, even in harder languages such as English, the degradation is still often acceptable [31], especially if the ultimate goal is keyword spotting.

3.3.1 Pronunciation Mixture Models (PMM)

Another way to reduce the amount of expert knowledge required to generate a lexicon is by learning from speech recordings in a data-driven manner. Mcgraw *et al.*. proposed Pronunciation Mixture Models which generate a more robust lexicon from an existing lexicon or can learn pronunciations for new words [56]. The PMM can be summarized as follows:

Algorithm 2 Pronunciation Mixture Models

1: Learn a Grapheme-to-Phoneme (G2P) model based on the existing lexicon.
2: Over generate the pronunciations of each word (including words not in the lexicon) to create a new lexicon.
3: Use the new lexicon to generate forced alignments on the training data.
4: Counts the expected counts for each pronunciation to estimate the weight for each pronunciations.

A G2P model is a model that predicts the pronunciation based on the graphemic representation of the word. For our application we use the model proposed in [3] which is a model based on letter n-grams.

3.3.2 PMM Experiments

We conducted experiments to evaluate the effectiveness of the PMM to generate pronunciation for new words. We trained systems on the FLP condition on Turkish and Pashto. However, instead of using the FLP lexicon, we started with the LLP lexicon instead. The LLP lexicon was used to train a G2P, which was then used to generate pronunciations for all words that exists in the FLP. A PMM was then applied to relearn pronunciation weights. Ultimately, we only kept the top 5 pronunciations for each word. The acoustic model setup was very similar to the previous experiments except for the features, which are BN features generated by the Brno Institute of Technology as outlined in [41] for a more robust system.

Table 3.3: WER comparison between systems trained using the FLP lexicon and the PMM lexicon.

Language	FLP lexicon baseline	PMM lexicon
Turkish	51.4%	51.3%
Pashto	53.1%	53.0%

Table 3.3 summarizes the WER results on the PMM experiments. As shown, the PMM model works just as well as the FLP lexicon even though it starts with only a subset of the full lexicon. Further analysis shows that the original lexicon sometimes

includes too many rare pronunciations which can hurt performance. Figure 3-1 shows examples of the expert pronunciations and the ones learned by the PMM, which all seem reasonable. This experiment shows that we can reduce the amount of expert knowledge required in generating the pronunciation lexicon. However, we still need a starting lexicon for this setup to work. On the other hand, the graphemic lexicon which requires no existing lexicon or knowledge about the phoneme inventory can offer comparable results as described previously.

Trafolardan

Method	Weight	Pronunciation
Baseline		t 1 r a f o 5 a r d a n
PMM	0.78	t r a f o 5 a r d a n
PMM	0.22	t 1 r a f o 5 a r d a n

Ozan

Method	Weight	Pronunciation
Baseline		o z a n
PMM	0.60	o z a n
PMM	0.21	a z a n
PMM	0.19	o: z a n

şüpheleniyorum

Method	Weight	Pronunciation
Baseline		S y p e l e n i j o r u m
Baseline		S y p h e l e n i j o m
Baseline		S y p h e l e n i j o r u m
PMM	1.0	S y p e l e n i j o m

ولي

Method	Weight	Pronunciation
Baseline		w a l i:
PMM	0.41	w a l i:
PMM	0.39	w @ l i:
PMM	0.20	w a l e

Figure 3-1: Examples of the pronunciation of new words learned by the PMM. Baseline pronunciations are pronunciations from the FLP lexicon.

We would like to end this section with a note about logographic languages such as Cantonese. It is not possible to use a graphemic lexicon for Cantonese. However, there are workarounds such as using pinyin to represent the characters, or using a small starting lexicon to learn the pronunciation of the rest of the characters from transcribed data in a data-driven manner such as the PMM we described or the method described in [9].

3.4 OOV Handling

Another issue that arises from limited transcribed data is the OOVs which we discussed in Section 2.2.3. The smaller amount of training text comes with a smaller

vocabulary size, and thus higher OOV rate. Table 3.4 shows a plot of how much the vocabulary size and the keyword OOV rate varies as we increase the amount of transcribed training data. The statistics were computed using the evaluation keywords. Here we define keyword OOV rate as follows:

$$\text{Keyword OOV Rate} = 1 - \frac{\#\text{ of keywords with at least one OOV word}}{\#\text{ of keywords}} \quad (3.1)$$

Table 3.4: Keyword OOV rate for various amounts of training data in four languages.

Amount of Training data	Cebuano	Kurmanji	Telugu	Swahili
40 hours (FLP)	11.84%	7.47%	13.93%	11.80%
10 hours (VLP)	37.27%	27.53%	33.14%	39.60%
3 hours (VLLP)	50.57%	41.68%	47.92%	54.81%

At 3 hours of training the data, the keyword OOV rate is around 50%. Even with a rather large amount of transcribed data the OOV rate still remains at 10%. In a naive implementation, performing KWS on an OOV term will return no results. Thus, it is important to be able to handle OOV terms even in a setup with a higher amount of transcription. For the rest of this section, we will focus mainly on the VLLP which has the highest OOV rate.

3.4.1 Subwords

Often times words can be broken down into smaller units, such as syllables or morphemes. Narasimhan *et al.*. proposed a method to use subword units for KWS [60]. Subwords can be used as the vocabulary words for decoding instead of words. The pronunciations of the subwords can either be generated by a G2P for the case of a phonetic system or be entirely graphemic. The LM can be trained by splitting the original text into subwords. Using these units, we might be able to represent OOV terms with known units. Table 3.5 summarizes the keyword OOV rate for different subword units on the VLLP condition.

Table 3.5: Keyword OOV rate using different subword units on the VLLP condition.

decoding unit	Cebuano	Kurmanji	Telugu	Swahili
word	50.57%	41.68%	47.92%	54.81%
morph	17.21%	9.02%	11.46%	9.63%
syllable	7.62%	1.59%	4.14%	8.31%

As shown in the table, subword units can reduce the effective OOV rate significantly. Syllable units are more effective in reducing keyword OOV rate, since they are shorter, making them more flexible. The keyword OOV rate can be as low as 2% as in the case for Kurmanji.

3.4.2 Phonetic Matching

As shown in Table 3.5, there are still some OOV terms left even after using subwords. These are often names or foreign terms. As a result we also need another granularity to search for OOV terms. Phonetic matching is often used as a final method to be able to catch all the OOV terms.

First, the keyword is expanded into a list of possible phone sequence using a G2P (or converted directly to a sequence of graphemes). The list is often expanded further using a phoneme confusion matrix which specifies which phonemes are easily mistaken for which other phonemes by the recognizer. The confusion matrix can be constructed by aligning the recognition results with the ground truth phonetic sequence. The confusion matrix can also include insertion and deletion possibilities. This process will give a new set of phone sequences as the search terms. This expansion is often referred to as “Fuzzy Matching”.

On the lattice side, we first generate a normal word or subword lattice [48]. The lattice is then broken down into a phonetic lattice. This works better than a phonetic decoder since there is a word language model that helps constrain the recognizer.

3.4.3 OOV Handling Experiments

To evaluate the effectiveness of these techniques in handling OOV keywords. We conducted KWS experiments on Swahili VLLP, which is the language for the OpenKWS 2015 evaluation. We trained a CD 3-state HMM-GMM recognizer using the graphemic lexicon in the same manner as described in Section 3.2.2. This baseline ASR system operates at 68.2% WER. The morphemes were generated using the Morphological Chains model [60]. The syllables were generate using the method described in [13]. The method assigns each grapheme a label of either consonant or vowel. A syllable can then be assigned based on the vowel locations.

For system combination of the KWS outputs, we used a simple method of averaging the scores from multiple systems if the hypothesis overlapped. We kept only the top 100 scoring hypothesis for each keyword to reduce the amount of false alarms.

Table 3.6: MTWV results on Swahili's dev set using different subword units.

	IV Keywords	OOV Keywords	All keywords
Word	0.2745	0.0000	0.1736
Syllable	0.1833	0.1500	0.1701
Morpheme	0.2512	0.1721	0.2205
Phoneme	0.2349	0.0076	0.1512
Combination	0.3469	0.3546	0.3449

Table 3.6 summarizes the effect of using the subword and phonetic search. As shown in the table, for IV terms, word systems usually perform the best. However, for OOV keywords, the system cannot produce any hypothesis, thereby it receives a MTWV of 0. Comparing the results of syllables with morphemes shows that morpheme-based lattices outperformed grapheme-based for both IV and OOV keywords. This is because grapheme are generally longer units, and thus offers more reliable scores. The system based on phonemes perform relatively well in IV keywords, but poorly on OOV keywords due to generating too many false alarms. Finally, these systems are highly complimentary. The combination of the four systems gives another 0.07 increase for IV and 0.17 for OOV over the best performing system.

Note that even more sophisticated techniques such as searching for an acoustically

similar sounding word that exists in the vocabulary instead of the original OOV keyword [8] can improve the MTWV on OOV keywords even further.

3.5 Web Data Usage

Web crawling is often used to acquire a large amount of text data to train stronger LMs. Web data also gives access to a larger set of vocabulary terms which helps reduce the OOV rate. However, web crawling is not a trivial task for low resource languages. Often times web resources are corrupted with other languages such as English for languages which use Roman alphabets. It is important to clean and filter web data before using it. The details of how such a process can be done can be found in [99]. The amount of text web data after filtering are summarized in Table 3.7.

Table 3.7: Text data available for four languages.

	Cebuano	Kurmanji	Telugu	Swahili
Word count	38M	49M	6M	16M

Using web data, we can add the most frequent words into the vocabulary. Figure 3-2 and 3-3 show the reduction in keyword OOV rate and transcription OOV rate as we add more words into the vocabulary. The transcription OOV rate is defined as follows:

$$\text{Transcription OOV Rate} = \frac{\#\text{ of OOV words in the transcription}}{\#\text{ of words in the transcription}} \tag{3.2}$$

As shown from the figures, web data augmentation can greatly reduce the OOV rate. By the time 30,000 words are added, both OOV rates are almost halved in all languages except Telugu. Telugu with its rich morphology has a larger vocabulary, so the OOV rate remains high even after 50,000 words are added.

Since written text is usually different from conversational speech, it is often the case that the LM perplexity barely improves over the original LM without web data. Table 3.8 shows the LM perplexity on the dev data with and without web data. Here

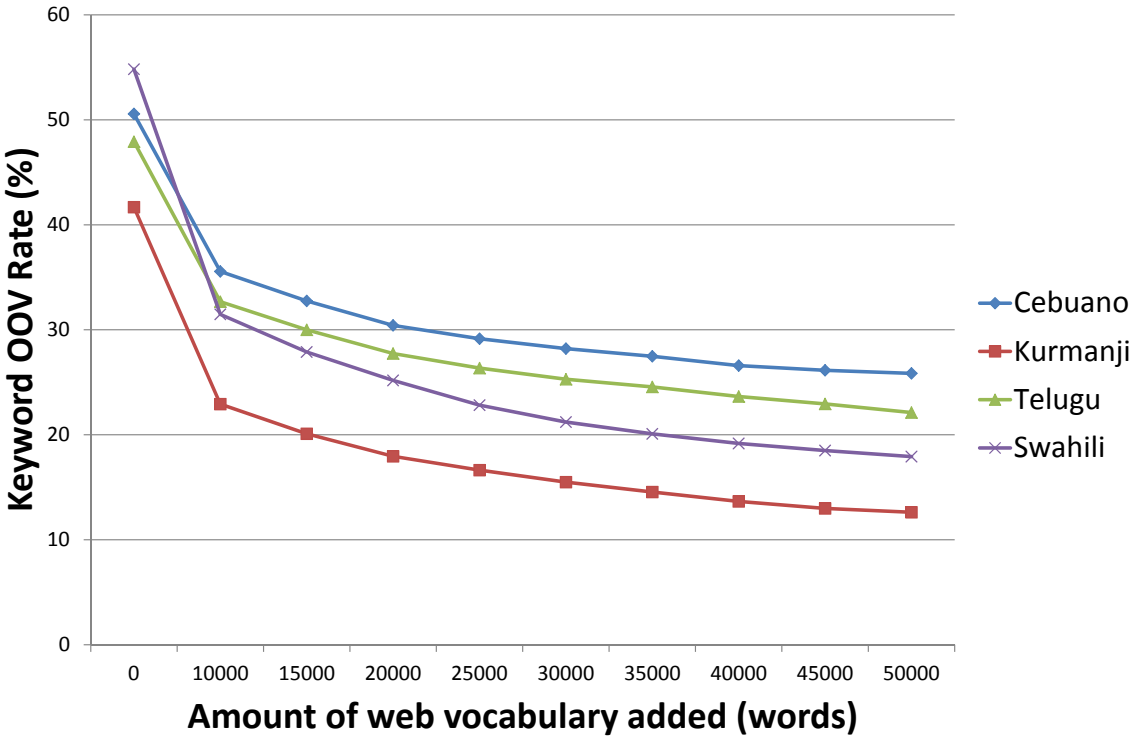

Figure 3-2: Transcription OOV rate as a function of the amount of vocabulary added from web data. The starting vocab is the VLLP condition.

we added only the 30,000 most frequent words into the vocabulary. We also include the LM interpolation weight for the VLLP text when we include an LM trained on web data. As shown, the perplexity is better with the web LM. However, because of the larger vocabulary size, the perplexity is often worse compared to the original VLLP LM. The interpolation coefficient for the VLLP LMs are all very high, signalling a mismatch between conversational speech and text from the web.

Table 3.8: Perplexity of LMs trained with and without web data.

	Cebuano	Kurmanji	Telugu	Swahili
LM without web data	132	168	312	246
LM with web vocab	185	250	583	382
LM with web vocab and text	170	220	485	307
Interpolation coefficient for VLLP LM	0.81	0.70	0.51	0.71

We then compare the LMs in the context of ASR. We built simple baselines with PLP, F_0, and PoV features using the setup described in Section 3.2.2. Table

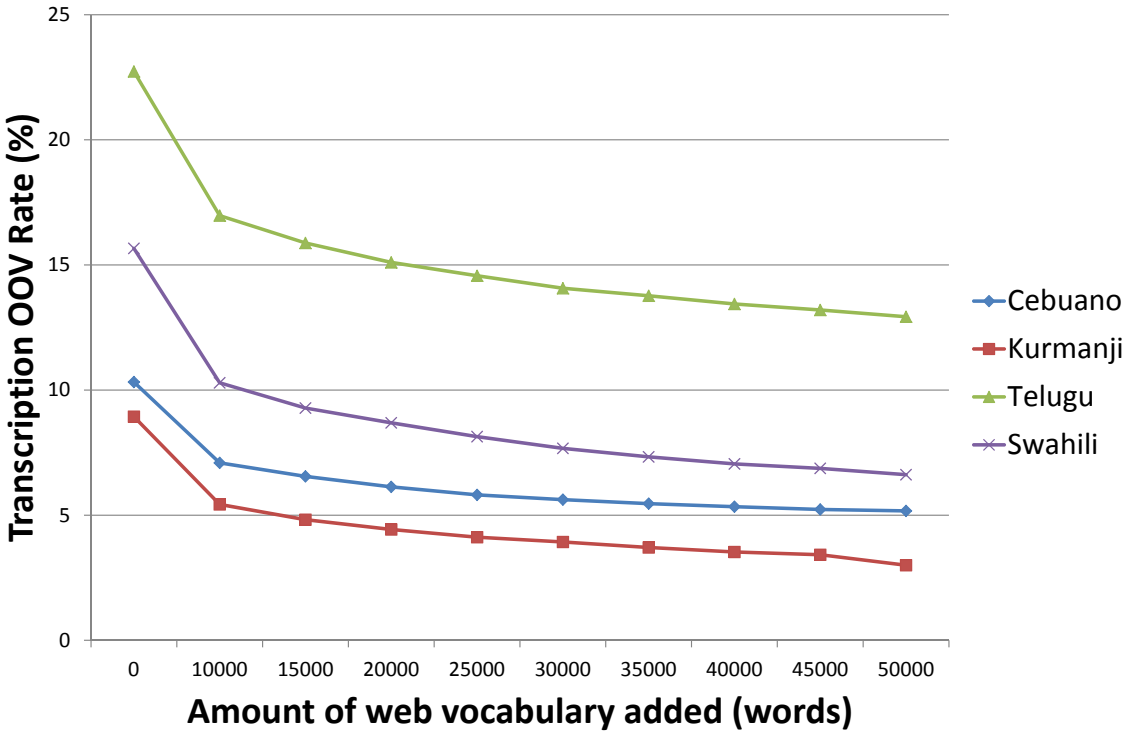

Figure 3-3: Keyword OOV rate as a function of the amount of vocabulary added from web data. The starting vocab is the VLLP condition.

Table 3.9: WER and MTWV comparisons of models with and without web data.

	Cebuano	Kurmanji	Telugu	Swahili
WER without web data	75.7%	85.5%	86.3%	68.2%
WER with web data	75.9%	86.4%	86.9%	66.3%
MTWV without web data	0.0869	0.0133	0.0166	0.1736
MTWV with web data	0.1312	0.0178	0.0215	0.2525

3.9 compares the WER and MTWV between systems built with and without web data on the dev set. As shown in the table, the MTWV always improves with web data. However, only Swahili got an improvement in WER after adding web data. In general, augmenting with web data will hurt WER because of the language mismatch. However, the richer vocabulary helps to improve KWS because it performs better when keywords are in-vocabulary.

From here on, whenever we use data augmentation from the web, we will include only the most frequent 30,000 words. This is chosen from the trade-off between

computation efficiency and OOV reduction rate.

3.6 Summary

In this chapter, we described many techniques that can be used to help improve ASR and KWS performance for low-resource languages. We started from exploring F_0 features that not only help tonal languages but also improve performance on non tonal languages. Due to its robustness, we will include F_0 and PoV features in all of our experiments from this point.

For lexicons, there are two possible approaches depending on the available resource. In the case where there is some initial lexicon, the PMM can be used to learn pronunciation of new words. The learned lexicon works as well as an expert lexicon for the languages we tried. In the case where there is no lexicon at all, we found graphemic lexicons to be an acceptable solution.

Finally, we looked into the problem of limited text data, especially the problem of OOVs which can greatly effect KWS spotting performance. We found that using subwords such as morphemes and syllables can greatly reduce the keyword OOV rate and improve KWS results on both IV and OOV keywords. Using web data also leads to an improvement in KWS due the reduction in OOV rate.

Chapter 4

Basic Multilingual Systems

4.1 Introduction

Using multilingual acoustic modeling to combat the insufficient data problem has been studied by many researchers. However, due to the limitation of available corpora most research had been forced to use rich resource languages, such as work by Burget *et al.*. [6] which used English, Spanish, and German which are languages that are closely related, so the conclusions might not apply in general. Besides the Babel corpus, there exists another smaller corpus that is tailored specifically for multilingual ASR research, the GlobalPhone corpus [79]. In this chapter, we will explore, as a baseline, the methods used widely on the GlobalPhone corpus, namely training a shared acoustic model on a shared phoneset [81].

4.2 Model Sharing Using Shared Phonemes

The paradigm used in many of the experiments conducted on GlobalPhone (or previous research in general) is based on the assumption that phonemes across different languages are similar, so the phonemes can be used as a language independent unit to share acoustic training examples between languages. In other words, there exists a global phoneset that contains all the phonesets from each language and is highly shared between the languages. Examples of such phonesets can be based on the In-

ternational Phonetic Alphabet (IPA) symbols [38] or other schemes such as Speech Assessment Methods Phonetic Alphabet (SAMPA) [96] or Worldbet [34].

Thus, one method to train a multilingual speech recognizer is to first map all the phones of each language into a common phoneset. The phoneset should also be compact in the sense that languages should share the same symbols as much as possible. Using a shared pool of phonemes will give more training examples to better estimate models for each phoneme. The hope is that the recognizer trained on this global phoneset would be a language independent acoustic model that can be used on a target language not seen in training. To use the recognizer on a target language, one would just construct the lexicon of the target language based on the global phoneset, and tie it with the language-independent acoustic model.

Figure 4-1 depicts the training and testing process of the language independent acoustic model using a global phoneset

4.3 Global Phoneset Experiments

We conducted experiments to see the effectiveness of this framework on the Babel Corpus. We chose the LLP datasets of Turkish, Pashto, Tagalog, and Cantonese as the source languages. These are the languages that were first available in the Babel corpus. For the target language we picked Vietnamese, which is linguistically similar to Cantonese. Vietnamese was also the language for the OpenKWS 2013 evaluation. We trained a simple 3 state HMM-GMM recognizer using features consisting of 13 dimensional PLP, F_0, and probability of voicing as explained in Section 3.2. We also include the Δ and Δ^2 of all the features making it 45 dimensional in total.

We created a pool of 67 phonemes based on SAMPA phones. Language specific phonemes that did not appear in the global phoneset were either mapped to an existing phone or split into the one of the global phones, e.g. phoneme /kw/ in Cantonese was split into /k/ and /w/. Tones are ignored. For the full list of the mappings see Appendix A.

Table 4.1 shows the recognition results of the model when tested on the four source

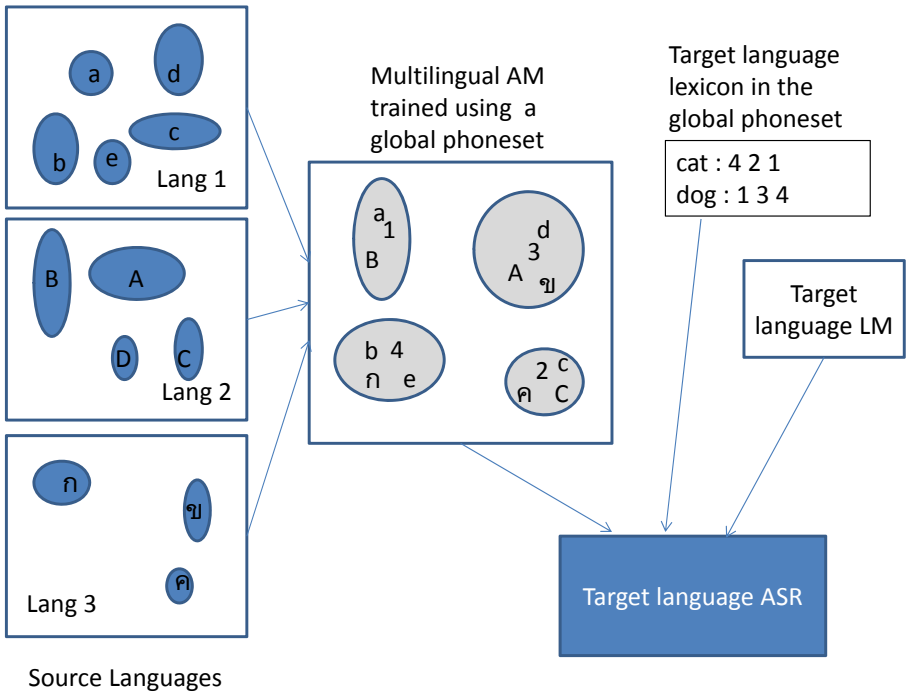

Figure 4-1: A simplified view of how a multilingual ASR can be trained and applied to a target language based on using a common global phoneset. The numbers represent phones in the global phoneset, while the different characters represent the phonemes in different languages. The multilingual AM can be trained by mapping different phonemes to the same global phoneme. Finally, an ASR system can be built for the target language by using a lexicon that uses the global phoneset.

languages. We also include the performance of the system trained on the original lexicon provided in the corpus. From the results, there is no real difference between using the original lexicon and the mapped lexicon based on the global phoneset, which means our mapping is working correctly. However, there are WER degradations of around 3% in the multilingual training for all languages. This is consistent with the work in [81]. Since the multilingual training adds additional confusibility to the model, the model degrades for the source languages.

Table 4.1: WER on the source languages using for the model trained on the multilingual phoneset. The Babel lexicon column indicates systems trained on the original phoneset, while the Multilingual lexicon column includes system that uses the global phone mapping. Monolingual GMM indicates that the system is trained on one language, while the multilingual GMM system is trained on all four languages. For Monolingual GMM systems, the language used for training is the same as the one used to evaluate the system.

	Babel lexicon	Multilingual lexicon	
	Monolingual GMM	Monolingual GMM	Multilingual GMM
Cantonese	76.1%	76.0%	77.8%
Pashto	79.6%	79.5%	82.0%
Turkish	80.8%	80.9%	84.4%
Tagalog	81.5%	81.8%	84.8%

Table 4.2 shows the recognition results of the model when tested on the target language, Vietnamese. As a baseline, we also include the performance for systems trained on only Vietnamese data. From the table, the multilingual acoustic model performs much worse than the monolingual baseline. Upon further analysis of the WER, most of the errors are deletion errors. The multilingual recognizer tends to generate ≪foreign≫ and ≪unintelligible≫ tokens rather than actual words when tested on Vietnamese. The work in [81] applied two additional training passes to adapt the multilingual model for some gain in performance. Our model, however, degrades when the same adaptation is applied. This might be because the original WER is too high.

Table 4.2: WER on the target language, Vietnamese, using the model trained on the multilingual phoneset. The multilingual GMM is trained on the source languages which do not include Vietnamese.

	Babel lexicon	Multilingual lexicon	
	Monolingual GMM	Monolignual GMM	Multilingual GMM
Vietnamese	80.7%	81.3%	95.2%
With Adaptation	n/a	n/a	96.2%

4.4 Analysis

The recordings in GlobalPhone are read speech which are carefully recorded in a very uniform recording conditions. However, the Babel corpus consists of conversational speech in noisy conditions which is inherently harder (The WER on GlobalPhone are mostly in the 30% WER range). Conversational speech causes the multilingual model to become too broad which causes the ≪foreign≫ and ≪unintelligible≫ models to dominate in mismatched conditions. Another important factor is the channel effect due to the different cellular carriers in each country. The mismatch in acoustic conditions can cause the models to perform poorly. Ragni *et al.*. showed that Cantonese's acoustic models are very different from other language [70]. We will also present evidence supporting channel effects in Section 6.4.3.

Note that mapping phonesets requires an existence of the lexicon and linguistic knowledge of the target language, which we might not have access to in certain scenarios. There are methods that can learn the mapping automatically [28], but a starting lexicon and a working recognizer in the target language are still required.

4.5 Summary

In this chapter, we applied a baseline method for training Multilingual ASR to the Babel corpus. A global phoneset is constructed to represent all possible speech phonemes. The lexicons are then mapped to the phoneset in order to train a single multilingual ASR using training data from multiple source languages. Unlike previous work which used the GlobalPhone corpus, we noticed a large degradation

using this technique due to the fact that recordings in Babel are less uniform due to the spontaneous nature and channel variations. These observations have also been confirmed by Knill *et al.* [45].

Chapter 5

Low-rank Stacked Bottleneck Architecture

5.1 Introduction

In Section 2.5 we described two frameworks for incorporating DNNs in a recognizer, namely the hybrid DNN-HMM approach, and the Tandem approach. The Tandem approach uses a DNN as a feature extractor for a standard GMM-HMM recognizer. This approach has the benefit of being able to use existing techniques developed for the GMM-HMM framework such as discriminative training, or speaker adaptation. Having access to robust features also makes it easy to work in a low resource setting. One can easily train the feature extractor on a high-resource language and use it on a low-resource language. However, there usually exist a gap in performance between the Tandem approach and the hybrid approach, since the existence of the low-dimensional bottleneck layer reduces the effectiveness of the DNN. In this chapter, we propose a method to improve existing Tandem approaches to be more in line with the hybrid approach[1].

[1]Many of the findings in this Chapter were published in [101].

5.2 Model Description

Before we describe the proposed method, we provide an overview of two related efforts that inspired our BN architecture: low-rank matrix factorization for DNN weights, and the Stacked Bottleneck (SBN) framework.

5.2.1 Low-rank Matrix Factorization

The left side of Figure 5-1 shows a typical ASR DNN architecture, where there is a stack of hidden layers followed by a softmax layer. In the figure, the nonlinear activation function $\mathtt{F}$ are Sigmoids functions. If the top hidden layer has $\mathtt{h}$ hidden units, and there are $\mathtt{s}$ targets labels, the weight matrix will be of size $\mathtt{h} * \mathtt{s}$. Following Sainath *et al.*, [76] we investigate a low-rank approximation to the weights of the softmax layer of the network. By considering the weights of the softmax layer as a matrix, we can factorize the weight matrix into two matrices of lower rank. As illustrated by the right side of Figure 5-1, this is done by replacing the usual softmax layer weights by a linear layer with a small number of hidden units followed by a softmax layer. More specifically, a new BN output layer with $\mathtt{r}$ linear hidden units is inserted into the last weight matrix with a hidden layer of size $\mathtt{h}$, and a softmax layer with $\mathtt{s}$ state posterior outputs. This changes the number of parameters from $\mathtt{h} * \mathtt{s}$ to $\mathtt{r} * (\mathtt{h} + \mathtt{s})$. Notice that there is no non-linearity for this BN output layer. Instead of using this structure for hybrid DNNs, we use it for extracting BN features. There are two benefits of using this method. First, it ensures the best achievable frame accuracy even with a relatively small $\mathtt{r}$. Second, the linearity of the outputs for the BN layer prevents the loss of information when we treat the DNN as a feature extractor.

5.2.2 Stacked Bottleneck (SBN) Features

The idea of using hierarchical processing of DNNs has been explored by several researchers. Valente *et al.* uses a second NN to help correct the posterior outputs of the first DNN by feeding it a different set of features [91]. For low-resource languages, SBN features have shown promising results in [41]. One argument for the the used of

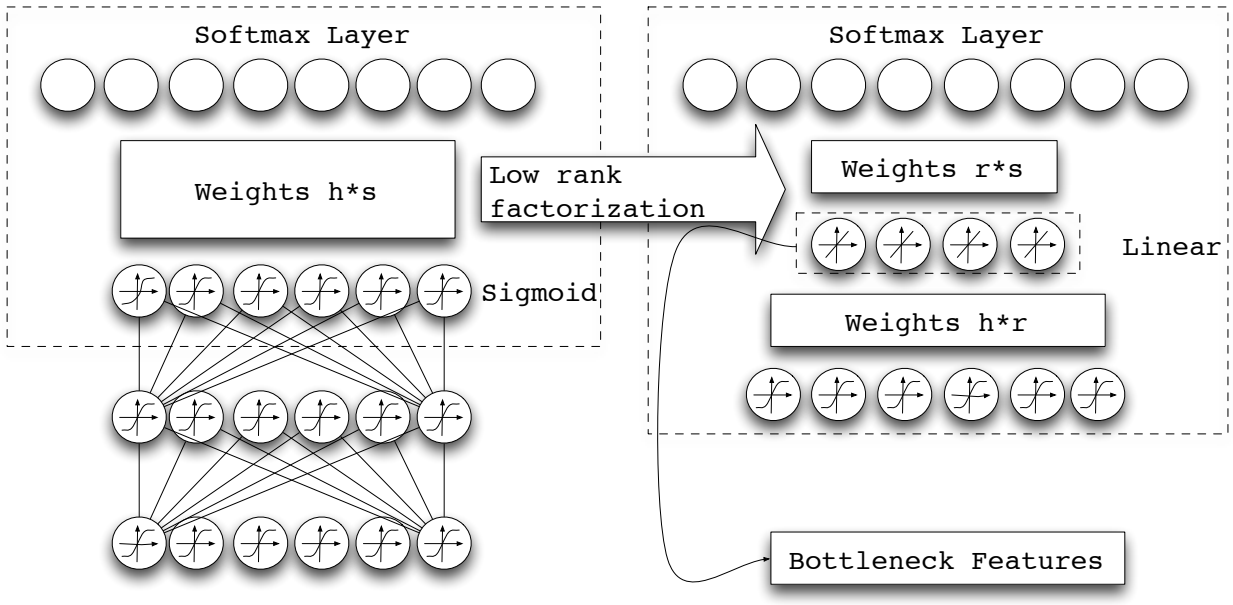

Figure 5-1: Diagram of the low-rank factorized DNN. The left side of the picture represents a typical DNN with `h` hidden units and `s` target labels. The right side of the picture shows the low-rank bottleneck DNN. The final layer is now replaced by two set of weights with a linear activation function in between. Bottleneck features can be extracted from the DNN by taking the output of the linear activations.

these cascading structures is that they enable more information, such as additional context, to be used more effectively [40].

5.2.3 Low-Rank Stacked Bottleneck (LrSBN)

Figure 5-2 gives an overview of our proposed low-rank SBN (LrSBN) feature extraction framework. The BN outputs from the first DNN are concatenated with context expansion and fed to the second DNN. Context expansion is done by concatenating frames with time offsets $-10, -5, 0, 5, 10$. We always place the linear BN layer (for the low-rank factorization) in the *last* hidden layer instead of a sigmoid BN layer in the *middle* of the network such as ones in [40, 95]. Experiments in [76] have shown that the hidden layers do not have the same low-rank properties as the weights in the softmax layer. We also use tied-states as the output targets instead of CI targets.

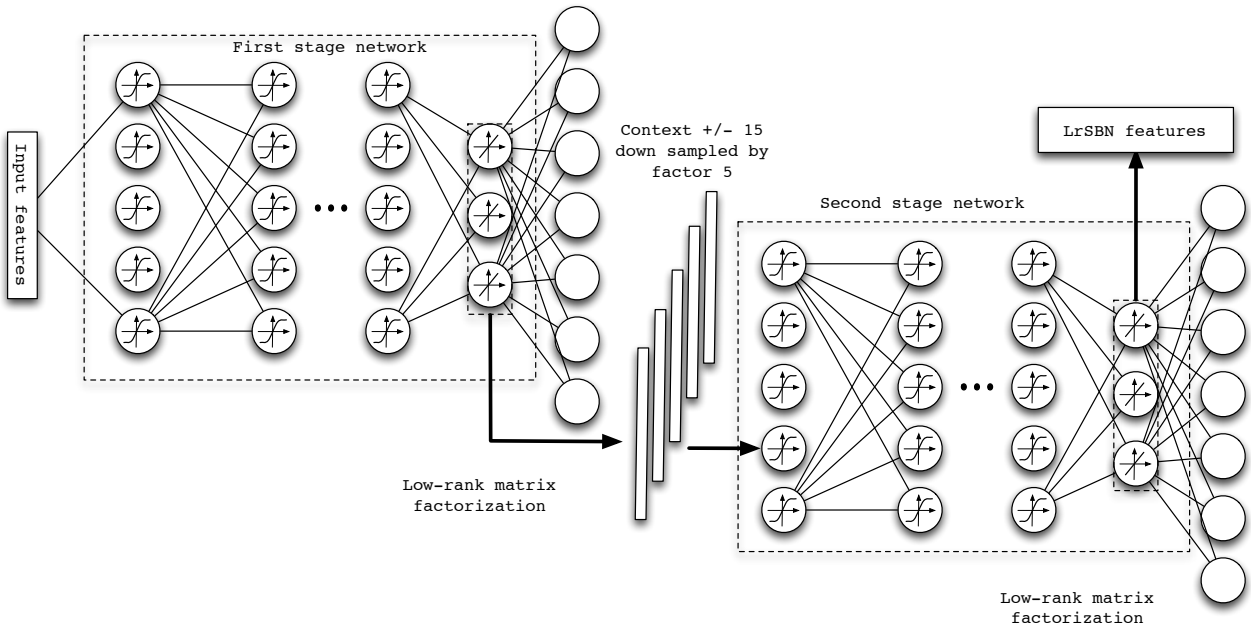

Figure 5-2: Diagram of the stacked bottleneck neural network feature extraction framework [101]. Two DNNs are combined together in a series. Starting from the left side of the picture, original input features are passed to the first low-rank BN network. The activations of the linear layer are extracted, and combined with activations computed from four other frames with time offsets -10,-5,+5,+10. The stacked feature is then used as input features to the second BN network. Finally, the LrSBN features can be extracted from the BN layer of the second DNN.

5.3 LrSBN Experimental Description

5.3.1 Baseline HMM Systems

Our baseline HMM systems were trained using the Kaldi ASR toolkit [65]. We used 13 dimensional PLP features concatenated with F_0 estimates and the PoV [89] as described in Section 3.2. Conversation-based mean and variance normalization was applied in both training and testing stages. The resulting 15-dimensional features were concatenated using ± 4 frames before and after the middle frame resulting in 135 dimensional vectors. LDA and MLLT [21] were applied to reduce the dimensionality and orthogonalize the features. Finally, a global fMLLR [20] was applied to normalize inter-speaker variability. For acoustic modeling, we used phonetic decision-based tied-state triphone CD-HMMs with ~2500 states and 18 Gaussian components per state. Trigram language models were created from training data transcripts.

5.3.2 Baseline Hybrid DNN Systems

The hybrid DNN systems were also created using Kaldi [65]. The DNNs had 6 hidden layers. The output layer was a softmax layer with target outputs corresponding to CD-HMM states. The network inputs were the speaker adapted features from the CD-HMM baseline (both for training and test) and concatenated using ± 5 frames (the total size is $40 * 11 = 440$). We used 1024 hidden units for each hidden layer. The nonlinearities in the hidden layers were sigmoid functions, and the objective function is the cross-entropy criterion. The alignment of CD states to frames was derived from the CD-HMM baseline systems and remained fixed during training.

The DNN was pre-trained by restricted Boltzmann machines and fine-tuned using the "new-bob" algorithm as described in Section 2.5.6. The initialization of the network and the learning rate followed the setting in [71]. We also perform sequence training with the sMBR criterion [94] to achieve the best possible results.

5.3.3 LrSBN systems

The input features for the first DNN (Figure 5-2) follows the method of [41] which extracts a kind of modulation frequency features. 23 critical-band energies are obtained from a Mel filterbank, with conversation-side-based mean subtraction. These features are augmented with F_0 and PoV features as described in Section 3.2. 11 consecutive frames are stacked together (110 ms). Each of the 23+2 dimensions is then multiplied by a Hamming window across time, and a Discrete Cosine Transform (DCT) is applied for dimensionality reduction. The 0th to 5th coefficients are retained, resulting in a feature of dimensionality $(23 + 2) * 6 = 150$. The feature extraction process is illustrated in Figure 5-3

The input features of the second DNN are the outputs of the BN layer from the first DNN. Context expansion is done by concatenating frames with time offsets $-10, -5, 0, 5, 10$. Thus, the overall time context seen by the second DNN is 31 frames (310 ms). Both DNNs use same setup of 5 hidden sigmoid layers and 1 linear BN layer. Both use tied-states as target outputs. The targets are generated by forced

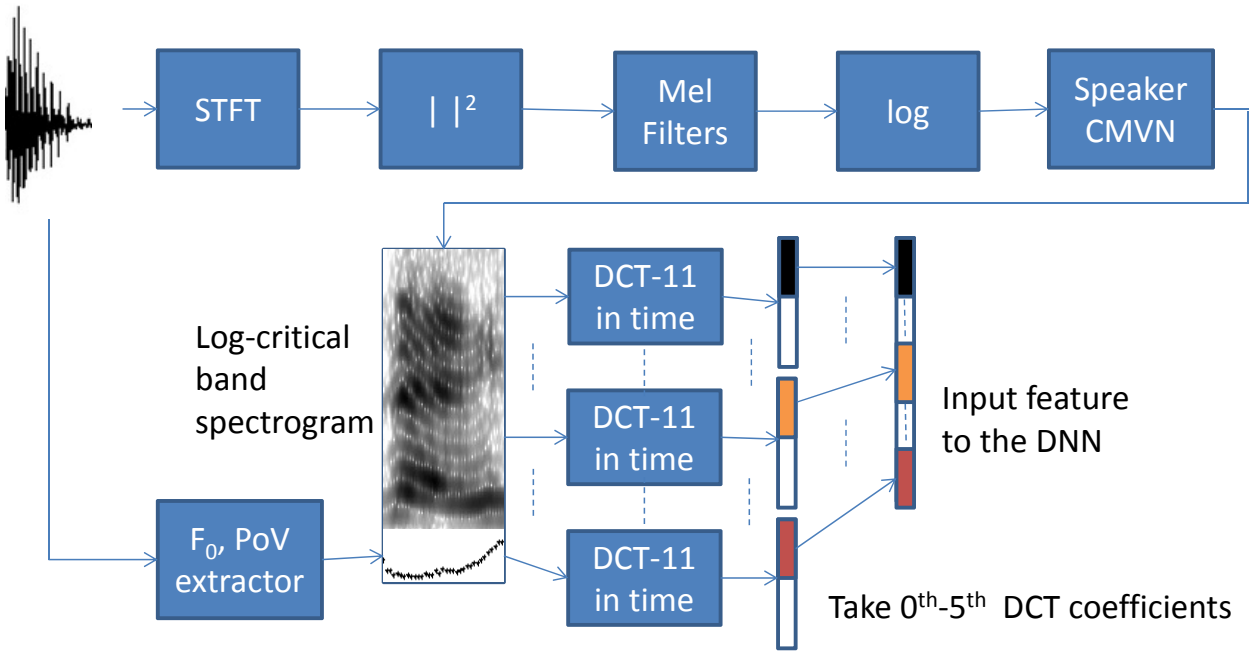

Figure 5-3: Diagram of the feature extraction used for the input to the DNN. The top portion of the figure follows a typical log-critical band spectrogram generation process. F_0 and PoV features are then augmented to the spectrogram. Finally, a DCT of size 11 are computed across time to extract modulation frequency information.

alignment from the HMM baseline. No pre-training is used. Finally, the raw BN outputs from the second DNN are whitened using a global PCA, and used as features for a conventional CD-HMM system.

5.4 Analysis of LrSBN Features

5.4.1 Context-independent (CI) vs Context-Dependent (CD) Labels

In [41], a DNN was trained to classify CI states. However, we have found, as have others [98], that using CD targets produces better results. Table 5.1 compares WERs between BN systems trained from CI versus CD labels on the Turkish LLP task. Note that we train CD GMM-HMMs using the extracted BN features whether or not the BN was trained using CI or CD targets. The PLP-based baseline for this task had

a WER of 75.0%. The first column in the table shows that if CD labels are used to train a single stage network, a 1.2% WER gain is obtained over CI labels. We also compare WER difference between single DNNs and a stacked architecture. A gain of 2.2% is obtained when CD labels are used to train the stacked network. Therefore, in all subsequent experiments described, we use only CD labels for SBN training.

Table 5.1: WER comparison on Turkish LLP for BN systems trained using CI or CD labels.

	# of target labels	Single DNN	Stacked DNN
CI targets	132	69.6%	68.8%
CD targets	1950	68.4%	66.6%

5.4.2 The Best Layer for Bottleneck Placement

Past research [76] has shown that DNN hidden layers do not all have the same low-rank properties. Following this observation, we compare the cross entropy per frame for different DNN configurations. Table 5.2 shows the average cross entropy (CE) per frame on the cross validation set for different BN placements on the Bengali LLP data. In all setups, the BN layers have 80 hidden units. As the table shows, putting a low-rank linear layer in the middle performs worse than a typical sigmoid BN layer. On the other hand, the low-rank softmax layer also has the lowest cross entropy per-frame. Thus, for all remaining experiments, we put the BN layer as the last hidden layer.

Table 5.2: DNN Comparisons of average CE per frame on Bengali LLP. 'Last' refers to putting the BN layer right before the softmax layer.

BN Type	Sigmoid layer		Low-rank linear layer	
BN Location	Middle	Last	Middle	Last
Avg. CE	0.253	0.250	0.257	**0.245**

5.4.3 Low-rank on the Softmax Layer

In order to determine whether the low-rank factorization on the softmax layer is necessary, we also evaluated the features generated by different activation functions on the Bengali Limited condition. The PLP-based GMM-HMM baseline for this task achieved 75.3% WER. In Table 5.3, we compare the results we achieve with BN features using a standard sigmoid on the softmax layer (SBN) with those obtained using the low-rank formulation (LrSBN). We also considered two BN derivations. In the first row of Table 5.3, we use the BN feature directly without any post-processing. In the second row, we do PCA on raw BN features, and reduce the dimensionality from 80 to 30. We then add Δ and Δ^2 BN features to model additional contextual information. It can be seen that for both conditions, the LrSBN achieves better performance.

Table 5.3: WER comparison on Bengali LLP between different BN DNN setups. SBN refers to a normal stacked bottleneck architecture with Sigmoid non-linearities. LrSBN refers to the proposed method which used a linear layer for the BN layer.

	SBN	LrSBN
raw BN	70.8%	69.2%
raw BN (PCA) + Δ + Δ^2	68.1%	67.2%

5.4.4 Results on Larger tasks and Different Languages

Further evaluation of the proposed method was performed on different languages with a speaker-adapted model. On this task, we compared the baseline GMM-HMM system, the hybrid DNN-HMM system, and the LrSBN system. The top of Table 5.4 shows that with standard ML training, an improvement of over 10% relative could be achieved when using LrSBNs. This result is even better than a hybrid DNN system that uses speaker-adapted input features. After speaker-adaptive training (fMLLR) and state-level minimum Bayes risk (sMBR) discriminative training [23] on the LrSBN features, the performance of the LrSBN system is similar to the hybrid DNN system with sequence training (SQ) in Bengali, while performing 0.6% better in the Assamese

case.

Table 5.4: WER comparisons between hybrid and tandem approaches on Bengali and Assamese LLP with different recognition setups. Three different sets of acoustic modeling techniques can be applied. The simplest system uses just the Maximum Likelihood (ML) training. LDA+MLLT feature transforms can be learned to train a stronger system. Finally, speaker adaptation (fMLLR) and discriminative training (sMBR) can be used for further improvements.

AM Training	ML	LDA+MLLT	fMLLR+sMBR
Bengali LLP			
PLP+F0	75.3%	74.4%	71.8%
DNN-HMM	n/a	n/a	68.0%
DNN-HMM+SQ	n/a	n/a	66.1%
LrSBN+Δ+Δ^2	67.2%	n/a	**66.0%**
Assamese LLP			
PLP+F0	74.6%	73.0%	70.5%
DNN-HMM	n/a	n/a	67.2%
DNN-HMM+SQ	n/a	n/a	65.7%
LrSBN+Δ+Δ^2	65.8%	n/a	**65.2%**

In addition to examining the limited condition training task, we also quantified the performance of the LrSBN features on the Bengali Full condition task. The WER comparisons are shown in Table 5.5. It can be seen the gain compared to the PLP baseline is even larger than for the limited condition case, with a 12.6% relative gain. It is also better than the Hybrid DNN system, improving the performance from 59.4% to 56.4%. Compared to the hybrid system with sequence training, the performance is still slightly better. Note that using sequence training has its disadvantages in terms of training time. For example, using a Tesla K20 GPU, an iteration of sequence training took up to 6 hours compared to the 40 minutes for cross entropy training. This can be an important constraint for the Babel project which emphasizes rapid system deployment. Using BN features also better lends itself to further improvements using standard techniques. Such improvements include better use of context information via discriminative training [64], and using speaker-adapted features as DNN inputs which we will describe in the following section.

Table 5.5: WER comparisons between hybrid and tandem approaches on Bengali FLP.

AM Training	ML	LDA+MLLT	fMLLR+sMBR
PLP+F0	69.2%	68.4%	64.5%
DNN-HMM	n/a	n/a	59.4%
DNN-HMM+SQ	n/a	n/a	55.6%
LrSBN+Δ+Δ^2	59.6%	n/a	**55.4%**

5.4.5 Speaker Adaptation on the First BN Output

Since the output of the first DNN can be used as an input to a regular GMM-HMM system, speaker adaptation techniques such as fMLLR [20] can also be applied on the first BN output for further gains.

Even though DNNs have the power to normalize the inputs from noise and other variabilities, they do so on a much smaller time scale, e.g. in frames. On the other hand, the transform learned in fMLLR is estimated over the entire 5 minutes recording, thereby offering complementary gains. Table 5.6 summarizes the improvement after we apply SAT on the first BN output, which seems to yield a gain of around 0.5% absolute improvement.

Table 5.6: WER after applying SAT on the first BN.

Language	LrSBN+Δ+Δ^2	with SAT
Bengali LLP	66.0%	65.4%
Bengali FLP	55.4%	55.0%
Assamese LLP	65.2%	64.9%

5.5 Multilingual Training of SBNs

Each DNN in the SBN can be trained in a multilingual fashion just like how one would train a multilingual DNN [28]. There are several methods for training multilingual DNNs. In [28, 44], a multilingual phoneset is created, and all the phonemes from the source languages are mapped to the set. However, in Chapter 4, we have found that using a multilingual phoneset might give worse results due to mismatches in

the data. The work in [28, 93] shows that a simpler scheme of concatenating each language outputs in the softmax layer can perform just as well. To do so, there are two different approaches depicted in Figure 5-4 and Figure 5-5

- **One softmax** The target output from each language are pooled together into one single big softmax layer. This has an effect of doing discrimination against all targets of the other language as well.

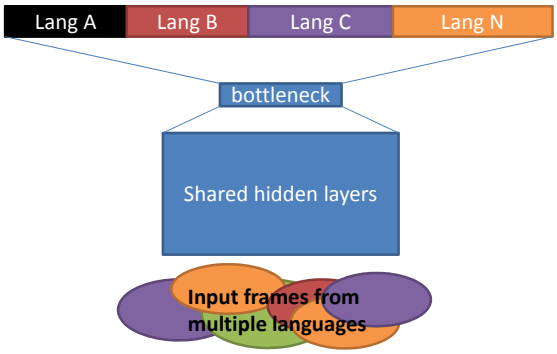

Figure 5-4: One softmax multilingual training. The target labels from multiple languages are combined into one large softmax layer.

- **Block softmax** The task is now considered as a multi-task training, where each language is considered as its own task with its own softmax with shared hidden layers. It is important that each mini-batch still includes all the languages used so that the parameters do not favor any particular language.

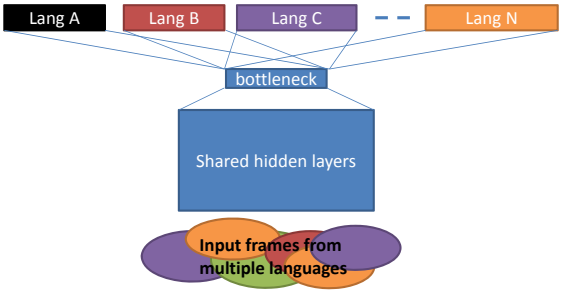

Figure 5-5: Block softmax multilingual training. Each language has its own separate softmax layers while the hidden layers are shared.

To compare the two training methods, we trained four different bottleneck DNNs. The first two were trained using the FLP of 5 languages, Cantonese, Pashto, Tagalog, Turkish, and Vietnamese, totalling 300 hours of training data. The other Two were trained using 11 languages, 500 hours of training data total. The additional 6 languages were Assamese, Bengali, Haitian, Lao, Tamil, and Zulu. We then used the DNNs to extract features to train recognizers on Cebuano VLLP. Table 5.7 summarizes the difference between each training methods.

Having more languages helps improve performance by 2% for both training methods, showing the power of language diversity. When comparing training methods, one softmax works as well as block softmax when the number of languages is small. Once there are more languages, block softmax does slightly better, possibly due the increased difficulty to differentiate between the same phonemes in different languages during training. However, one softmax is faster to train due to the simplicity of the model.

Table 5.7: WER comparison between one softmax and block softmax for training multilingual BN features. The target language is Cebuano VLLP.

Cebuano VLLP	One softmax	Block softmax
5 languages	72.4%	72.3%
11 languages	70.3%	70.0%

5.6 Summary

In this chapter we described the Low-rank Stacked Bottleneck (LrSBN) architecture. By placing a low-dimensional linear layer right before the softmax layer, we were able to exploit the low-rank properties of the softmax layer and extract better features. Using a hierarchical structure to exploit longer context frames also helped improve performance of the BN feature even further. Our LrSBN model was able to improve over the standard HMM-GMM models by 10% absolute WER, and over the hybrid DNN-HMM by 4% in the FLP condition. Since the publication of this work, using a linear layer for extracting bottleneck features has become the standard in the ASR

community [30, 45, 58, 74]. We have also looked into two different approaches for multilingual training of DNNs. We found block softmax, which train each language using its own softmax in a multi-task manner, to outperform simple concatenation of labels from each languages.

Chapter 6

Multilingual Transfer Learning Using Language Identification

6.1 Introduction

In the previous chapter, we described how one can use DNNs to learn better representations from acoustic data by using the LrSBN. Researchers have also used BN features to leverage out-of-domain resources, by training the DNN in a domain with rich resources and applying it on domains with limited resources, a technique often called transfer learning [85, 93, 95]. The resource used to train the DNN does not need to come from one language, but can come from a variety of languages. This so called multilingual training help make the DNN becomes more language independent, since it is trained on many kinds of phenomena that might exist in certain languages [93]. In [30], the target language data is also used for adaptation of the multilingual DNN by doing additional fine-tuning steps. These approaches not only alleviate the lack of training data, they also reduce the amount of time required to train DNNs for the target languages.

None of the work mentioned above addresses the issue of what to do when there are multiple source BN systems to choose from, i.e. having one BN system for each language. This is not an unrealistic scenario, as researchers often have multiple recognizers on hand. Since there is evidence that transfer learning works better when

the domains are closer, picking the correct source language for the target language would be more beneficial. Another related question is what are good languages to include when training multilingual DNNs. In this chapter, we will try to answer these two related questions.

In this chapter, we propose an effective Language Identification (LID) method to select possible candidate languages for transfer learning in the LrSBN framework. The same method can also be used to identify data that is most useful for multilingual training, which ultimately improves the ASR system on both transcription and KWS tasks[1].

6.2 Language Pair Transfer Learning

In this section, we motivate the potential benefits of language pair transfer learning, and whether these systems, especially the trained DNNs, can be used to facilitate the training of new target languages.

6.2.1 A Case Study on Assamese and Bengali

We start by looking at the best case scenario possible, namely the language pair of Assamese and Bengali. Assamese and Bengali are spoken in adjacent regions in India. They are known to be linguistically close, with overlapping phoneme inventories [72]. We used LLP Bengali and FLP Assamese as the target and source languages, respectively. The transfer learning was done by using the LrSBN architecture trained on FLP Assamese data to extract BN features for LLP Bengali. Tied-state triphone CD-HMMs, with 2500 states, and 18 Gaussian components per state, were used for acoustic modeling. We used the same input features as the ones described in Section 5.3.3. Discriminative training was done using the Minimum Bayes risk (MBR) criterion [23]. For language modeling, a trigram LM was learned from only the training data transcripts. We used the provided phonetic lexicon in this experiment.

[1]Many of the experiments in this chapter were first published in [100] and [10].

Table 6.1: WER on Bengali with different data usage scenarios. Assamese FLP data is used to train BN features to use in Bengali. The numbers under the BN and GMM columns refer to the amount of acoustic training data in hours used to train the bottleneck and the GMM, respectively.

System	Bengali data		Assamese data	WER
	BN	GMM	BN	(%)
(a) LLP PLP	10	10	0	71.8
(b) FLP PLP	62	62	0	64.5
(c) LLP BN	10	10	0	66.0
(d) LLP + FLP Assamese BN	0	10	61	64.6
(e) LLP + Adapted FLP Assamese BN	10	10	61	63.7
(f) LLP + FLP Bengali BN	62	10	0	61.6
(g) FLP BN	62	62	0	55.4

Instead of just using the DNNs trained on Assamese data, we also adapted them. This was done by replacing the original softmax layer with a randomly initialized Bengali softmax layer, and performing additional fine-tuning iterations on the Bengali LLP. Replacing the softmax layer completely eliminates the need to do phoneme mapping between languages, which can sometimes be complicated and does not guarantee good performance, as mentioned in Chapter 4. This adaptation process is equivalent to using the Assamese data to "pre-train" the Bengali network, which helps initialize the DNNs into a better starting point. With the better initialization the network typically converges in 5 epochs instead of the 10 epochs needed for a randomly initialized network. The adaptation is done on both DNNs, one at a time starting from the first DNN.

Table 6.1 summarizes the results of the experiments. As we have shown in the previous chapter, the LrSBN systems significantly outperform the standard PLP systems (a vs. c and b vs. g). The BN system trained on just 10 hours of data performs almost as good as the PLP system trained on 60 hours, which shows the effectiveness of having stronger features. The FLP BN system also has 10.6% lower WER compared to the LLP counterpart (c vs. g.).

Using the Assamese BN features trained on 60 hours of data improved the WER

Table 6.2: WER using between different language pairs. Numbers in parenthesis are Limited Monolingual BN baseline.

Target (Limited)	Source BN (Full)			
	Bengali	Assamese	Lao	Turkish
Bengali (66.0)		**63.7**	65.1	64.2
Assamese (65.2)	**61.2**		62.9	62.1
Lao (62.3)	**59.8**	60.1		60.0
Turkish (63.9)	**61.8**	63.1	63.3	

by 1.4% absolute over the 10 hour Bengali BN baseline system (d vs. c). Adapting the DNN to 10 hours of Bengali data reduces the WER even further to 63.7% (e). As an Oracle baseline, we also use a BN system trained on FLP Bengali to extract features for the LLP Bengali system (f). The WER of this setup is 61.6%, 4.4% lower than the baseline LLP BN system. Thus, transfer learning is able to improve the performance on 10 hours of Bengali data from 66.0% to 63.7%, which is around half of the gain of the oracle system where the BN features are trained on matched data.

6.2.2 Other Language Pairs

To look at the possibility of transfer learning in a broader scenario, where the languages are less similar, we expand our experiments to include two more languages, namely Lao and Turkish. Lao is a tonal language in the Tai-Kadai language family, while Turkish is in the Turkic language family. These two languages are unrelated to the Assamese and Bengali and, thus, chosen as contrastive languages. Table 6.2 summarizes the BN feature transfer learning WERs with target language adaptation on the four languages. As expected, the closest language pair of Assamese and Bengali seems to mutually benefit the most. Bengali also seems to be a good language in general for the other three languages.

6.2.3 Language Identification for Source Language Selection

Although the experiments in Section 6.2.2 are promising, we would like to avoid relying on expert knowledge on which language pairs would prove useful. Moreover,

Table 6.3: Average posteriors for the initial LID experiment.

Input frames	Predicted posteriors (Averaged)			
	Bengali	Assamese	Lao	Turkish
Bengali	0.57	**0.21**	0.09	0.13
Assamese	**0.21**	0.57	0.11	0.11
Lao	0.08	**0.11**	0.71	0.10
Turkish	**0.13**	0.12	0.10	0.65

in most cases language similarities are far from obvious from the limited pool of available source languages. The prospect of trying out all possible source languages might not be time efficient. We propose to use Language Identification (LID) as a way to determine which language to use as the source language. The LID system is a DNN with 2 hidden layers and 512 hidden units per layer for LID on the four languages. To compute the LID scores given a source language, we compute the posterior probabilities averaged across all frames in the target language. The LID scores should correspond to how close the target language is to each source language. The process is shown in Figure 6-1.

We randomly selected 90% of the Limited training sets for training the network. Unlike the DNN-based LID work in [100], we use the same input features as the ones described in Section 5.3.3. This is to make the LID DNN decide which languages are similar based on what the BN DNN would observe. We then use the DNN to classify the remaining 10% of the (held out) data. Table 6.3 summarizes the posteriors of each language, averaged across all frames. The closeness between Assamese and Bengali are again confirmed by the LID results, with average posteriors of 0.21. Turkish is also closest to Bengali, which is consistent with our previous experiment. Less similar pairs seem to also correspond to worse WERs in the previous experiment. The only language that does not follow the predicted trend seems to be Lao. However, the WER difference between using the closest language (in the LID sense) and the best possible outcome is only 0.3%. Thus, we believe that LID is a reasonable method to select a source language for transfer learning.

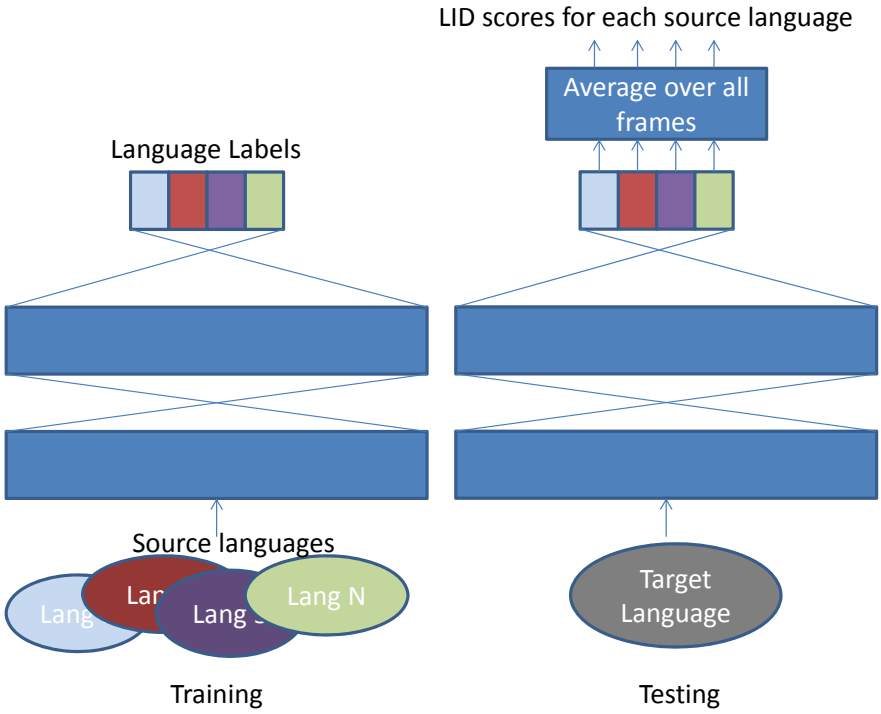

Figure 6-1: Training and testing of the LID DNN. In the training stage, frames from the source languages are used to train a DNN with language labels as the output target. At test time, the DNN is then used to compute posteriors scores which are then averaged over all frames.

6.3 Transfer Learning Experiments

In this section, we compare different multilingual strategies and their training time trade-off. For a stronger baseline, we modified the BN features described in Section 5.3.3 as follows: The filterbank inputs were processed with VTLN warping factors [77]. Speaker adaptation is also applied to the outputs of the first BN DNN before feeding it to the second BN DNN as described in Section 5.4.5. Cantonese, Turkish, Pashto, Tagalog and Vietnamese are chosen as source languages.

A multilingual stacked bottleneck DNN is trained on the Full condition of all source languages which consists of ~300 hours. The DNN training follows the One Softmax method described in Section 5.5. Language-specific speaker adaptation is then applied on the outputs of the first DNN. Similarly, the monolingual versions of all source languages are trained using this procedure.

6.3.1 Adaptation Strategies

Since there are two DNNs in the LrSBN architecture, several adaptation strategies are available. In [30], they observed that it is beneficial to adapt the first DNN. However, the second DNN should be either adapted, or trained from scratch using the target data only, depending on the target language. Thus, for our experiments, we always adapt the first DNN from either the multilingual DNN or the monolingual DNNs from the source languages. For the second DNN, the following approaches were considered:

Target only

The second DNN is trained from random initialization using only the target language data with the BN output from the first DNN as input features.

Multi

The second DNN is adapted from the corresponding multilingual DNN, as depicted in Figure 6-2.

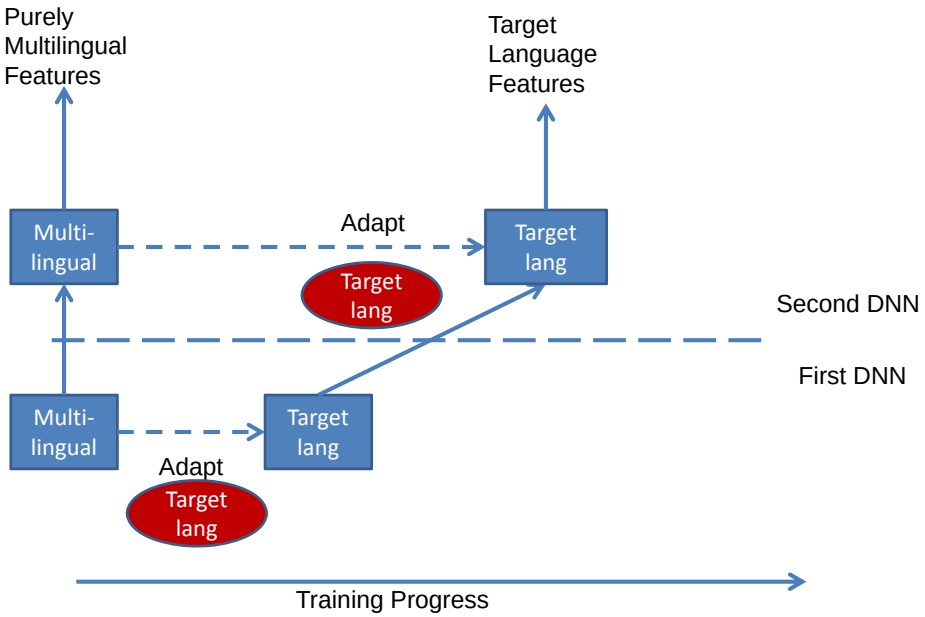

Figure 6-2: Adaptation of the SBN using the Multi method. The first and the second DNN are adapted using the data from the target language sequentially.

Mono

The second DNN is adapted from an existing monolingual DNN from a source language. The source language can either be selected at random, or selected using the LID scores. We depicted the case where we use the closest language according to the LID scores in Figure 6-3.

Mono re-train

Re-train from a random initialization the monolingual DNN using the features from the first multilingual DNN which is already adapted to the target language. Then, use the target language data to do adaptation as depicted in Figure 6-4. Just like the mono case, the source language can either be selected at random, or selected using the LID scores.

Multi re-train

Re-train from a random initialization the multilingual DNN using the features from the first multilingual DNN which is already adapted to the target language. Then,

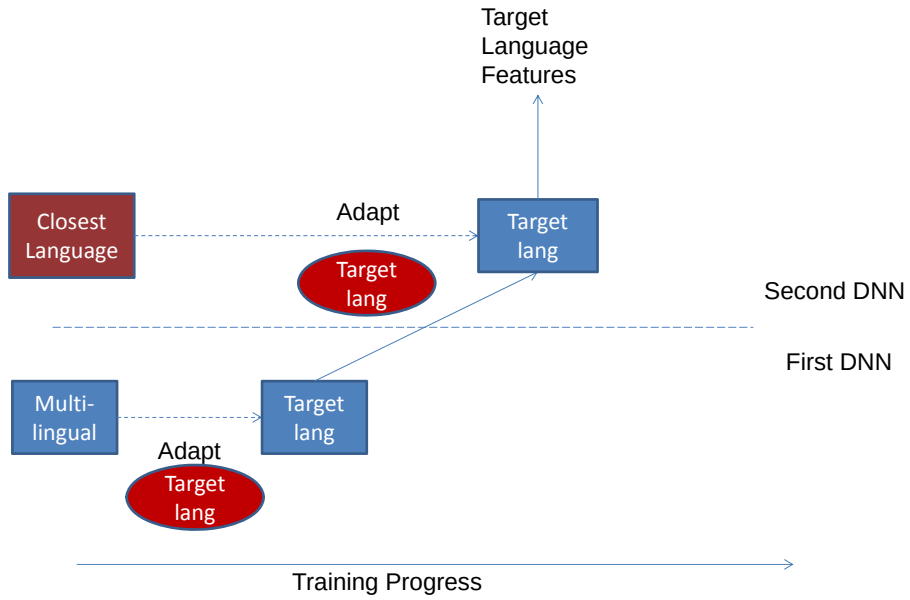

Figure 6-3: Adaptation of the SBN using the Mono re-train method using the LID scores. The first DNN is adpated from the multilingual first DNN. However, the second DNN is adapted from the DNN already trained on the closest language.

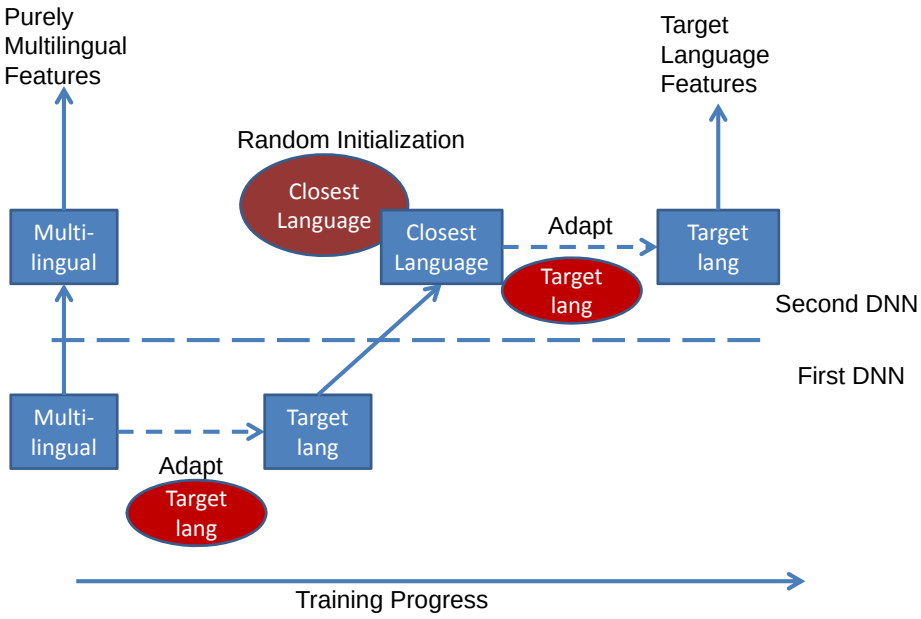

Figure 6-4: Adaptation of the SBN using the Mono method using the LID scores. The first DNN is adpated from a multilingual first DNN. BN features are extracted from the adapted DNN and used to train a new DNN on the closest language from random initialization. The DNN is finally adapted to the target language.

Table 6.4: LID posteriors of the 5 source languages languages.

Input frames	Predicted posteriors (Averaged)				
	Tagalog	Pashto	Turkish	Cantonese	Vietnamese
Lao	0.25	0.08	0.14	0.16	**0.37**
Assamese	**0.31**	0.18	0.19	0.06	0.26

use the target language data to do adaptation.

While the original multilingual DNN can be trained in advance, we do not consider multi re-train since re-training the multilingual DNN would take longer than one week. Re-training a Full condition's worth of data (mono re-train) would take around 13 hours. The adaptation of the DNNs with 10 hours of target language data (all methods) can be done in less than one hour.

6.3.2 LID-based adaptation Results

For our multilingual experiments we chose LLP Lao and LLP Assamese as our target languages. A DNN for LID was trained for the five source languages languages as described in Section 6.2.3. The average predicted posteriors from the Limited condition of the two target languages are summarized in Table 6.4. Lao is closest to Vietnamese, while Assamese is closest to Tagalog. Note that Pashto and Assamese fall under the same language family of Indo-Iranian, but Pashto is placed fourth in terms of LID similarity to Assamese.

Table 6.5 shows the results of the different adaptation strategies where LID denotes using the monolingual DNN from the closest source language language. For comparison, we also use Pashto as another possible source language. The baseline BN systems using only the target language data have a WER of 61.5% and 63.3% for Lao and Assamese, respectively. All the multilingual systems perform better than the baseline, showing the benefits of using additional resources to facilitate low-resource ASR. Adapting both DNNs improves the WER in all cases. As expected, using the closest language DNN performs better than Pashto (d vs. e). More importantly, using the monolingual DNN from the closest languages for the first and second DNNs

Table 6.5: WER on Lao and Assamese LLP using different adaptation strategies. Letters in parentheses denote the source language used for the monolingual DNNs; Tagalog (T), Pashto (P), Turkish (U), Cantonese (C), and Vietnamese (V)

	DNN for adaptation		WER (%)	
	1^{st} stage	2^{nd} stage	Lao	Assamese
a	target only	target only	61.5	63.3
b	multi	target only	59.0	61.2
c	multi	multi	57.5	59.4
d	multi	mono	58.0 (P)	60.1 (P)
e	multi	LID	57.5 (V)	59.4 (T)
f	LID	LID	56.8 (V)	59.3 (T)
g	LID	LID re-train	56.5 (V)	59.0 (T)
h	multi	LID re-train	56.0 (V)	58.5 (T)
i	multi	mono re-train	56.7 (P)	58.8 (P)

works slightly better than the multilingual counterparts (f vs. c). The multilingual DNN seems to help when coupled with re-training of the second DNN using the closest language (h), improving the WER by another 0.8% for both Lao and Assamese. This, however, comes with a slightly longer training time.

6.3.3 No Data like Similar Data

The experiments in the previous section show that the language identified as the closest identified language alone can achieve comparable performance to the combined multilingual training. Yet, the data used in the monolingual systems are strictly subsets of the multilingual data. This seems to imply that including other languages which are "further away" can hurt performance. To this end, we re-visit the Lao-Turkish (source-target) language pair, which provided the least performance gain in Section 6.2.2.

In the same way that LID can identify which language is closest to the target data, LID can also be used as a selection tool to determine which portion of the data is most useful for the target language. Suppose we train a LID DNN using all the data from the Limited condition of Lao and Turkish. Then, for BN DNN training,

Table 6.6: Effect of source data selection on Turkish LLP.

Amount	Lao data usage	WER (%)
0	None (Turkish only)	63.9
10hrs	LLP data	64.4
10hrs	Furthest utterances	66.5
10hrs	Closest utterances	63.8
10hrs	Closest frames	63.1
65 hrs	FLP data	63.3
32.5 hrs	Random utterances	64.0
32.5 hrs	Closest utterances	62.8
32.5 hrs	Closest frames	62.4

we select Lao data at either the frame or utterance level. At the utterance level, the frame posteriors are averaged across each utterance. The frames/utterances with highest posteriors are then used to train the source BN DNN for further adaptation.

Table 6.6 summarizes the different data selection strategies. Using the provided Limited Lao subset, the performance is even worse than the baseline with no Lao data at all. Unsurprisingly, the performance degrades even further if 10 hours of the furthest utterances from FLP Lao are used. Using 10 hours of the closest utterances, however, can achieve a WER of 63.8%, slightly better than the Limited Turkish baseline. Selecting based on frames gives a slightly better WER than utterance-based selection. The best performing system based on 10 hours of Lao data selects only the closest frames and attains an WER of 63.1%. With only one-sixth of the data, we do as well as if we had used the FLP Lao. Selecting the closest half of the frames yields a WER of 62.4%, a 0.9% absolute improvement.

From the results, having data that is similar to the target data seems to be more important than having more source data. This anecdotal observation seems to suggest that adequately robust BN features can be trained without much data, especially when the resulting DNNs are used as a starting point for adaptation. As less similar data would put the DNNs into worse initializations, perhaps we should exercise more care in selecting data for multilingual adaptation.

6.4 Frame Selection for SBN Training

In the previous section, we observed that selecting the closest subset from a particular source language can sometimes be more beneficial than using all of it. This offers some explanation to why the multilingual SBN can perform worse than the closest language setup. However, it has also been observed that in a multilingual setting, having more source languages usually helps due to better coverage of phonemes and acoustical phenomena, as well as the simple fact that there is more data [29, 44]. From these observations we propose an improvement over the closest language training scheme by selecting frames from all source languages that are closest to the target language to train the second DNN.

6.4.1 Frame Selection DNNs

To select the closest frames from the multilingual pool, we need a way to score and rank all the frames. We can do so by training N frame selection DNNs, one for each source-target pair, as shown in Figure 6-5. Each frame selection DNN is a two-class DNN where the training data are the frames from the source and target languages with their corresponding language labels. The score of a frame from any source language is then the posterior probability of that frame coming from the target language computed by using the corresponding DNN. Although each frame selection DNN is trained independently from the rest of the source languages, we observe that the distribution in the rankings of all source language frames correlates well with the scores given by the LID DNN; the languages with higher LID scores have more highly ranked frames.

We train N frame selection DNNs for the ranking instead of one single DNN with $\mathsf{N}+2$ output labels (the sources, SIL, and the target) because the existence of a close language pair in the source pool can skew the ranking of the frames. For example, consider the case when the source languages are Assamese, Bengali, and Zulu, and the target language is Telugu. Assamese, Bengali, and Telugu are all Indian Languages, so we expect the frames from the Assamese and Bengali to have higher probabilities of

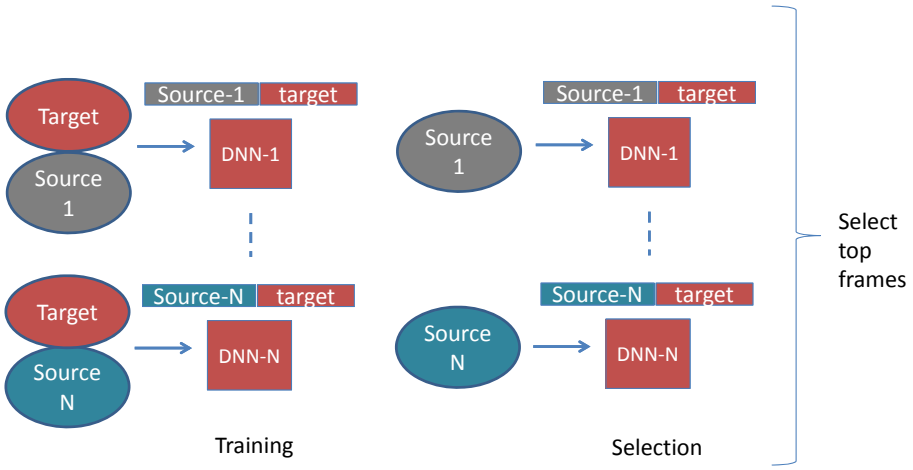

Figure 6-5: The frame selection process. N pair-wise DNNs are trained for each source language pair. Then, frames from each language can be ranked by using its corresponding DNN.

being Telugu than frames from Zulu. However, since Assamese and Bengali are very similar languages (more similar together than to Telugu), the posterior probability for an Assamese frame will mostly be biased towards Bengali. On the other hand, a Zulu frame would have no such effect and may have a higher posterior for Telugu.

After selecting the closest frames, the training for the DNN based on frame selection follows the same procedure the multi-retrain method discussed in Section 6.3.1. However, unlike re-training using the full multilingual pool which can potentially take a long time to train, we only need to use the subset of the multilingual data.

6.4.2 Frame Selection Experiments

We use 11 languages (FLP) as the source languages, namely Cantonese, Vietnamese, Tagalog, Pashto, Turkish, Bengali, Assamese, Zulu, Tamil, Haitian, and Lao. For target languages, we chose Cebuano, Telugu, and Swahili (VLLP). Just like in the transfer learning experiments in Section 6.3, we train a LID DNN using the 11 source languages, with one modification: we also add a silence target label in the softmax layers, so there are 12 labels total. The silence label eliminates the needs to filter out silence frames when computing posterior scores which helps in simplifying the process. We also train 11 frame selection DNNs for each target language, meaning

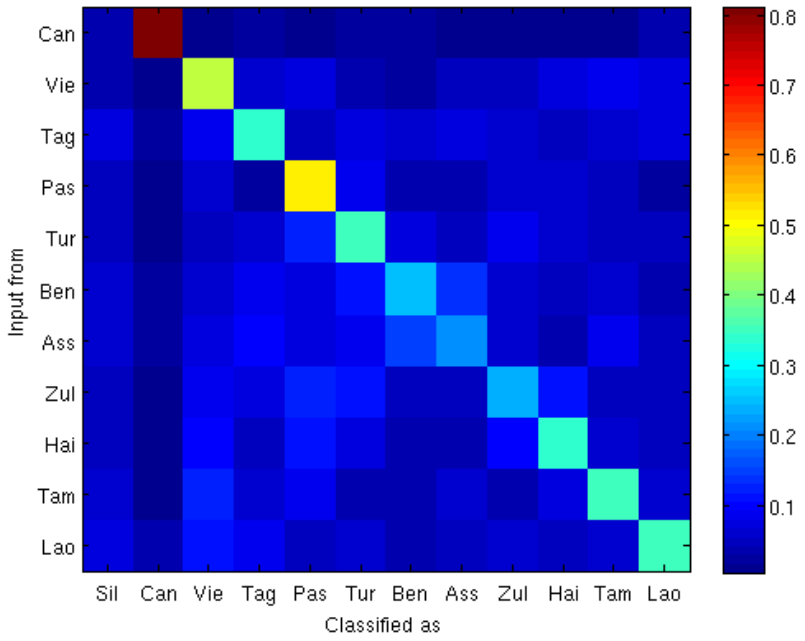

Figure 6-6: A heat map of the averaged posterior scores for the source languages. Each row indicates the misclassification from each language.

we trained 33 frame selection DNNs in total.

6.4.3 LID DNN Analysis

We start by analyzing the LID DNN in identifying the closeness between the source languages. We compute the LID scores using the dev set of the source language to generate a confusion matrix as depicted in Figure 6-6. As shown from the picture, the diagonals are always the highest value which indicates that the LID DNN is behaving as expected. The obvious Bengali-Assamese pair is also highly confusable. One interesting note is the fact that Cantonese is not confusable with any of the other source languages; even Vietnamese which is known to be similar linguistically. We believe this is due to a channel effect by the cellphone carriers in China. An independent study by IBM which tried to identify the closeness between languages using metrics derived from the GMM-HMM acoustic model trained on each language also pointed out that Cantonese is an outlier language [70].

To look at how the recording channel can effect the closeness between the data,

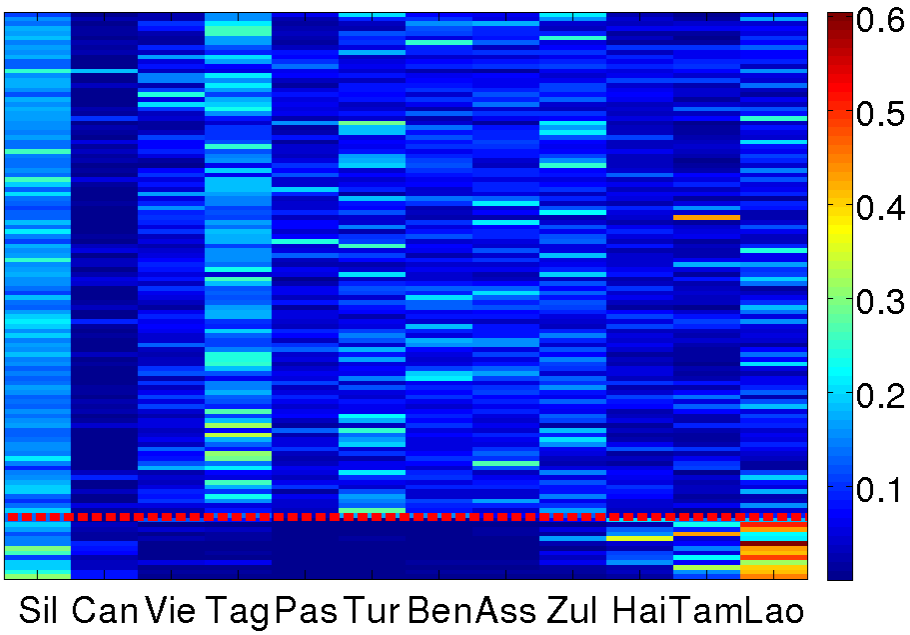

Figure 6-7: A heat map of the averaged posterior scores for each speaker from Cebuano. Each row in the figure refers to a speaker. Each column refers to the language output class. The speakers below the red dashed line are from wideband recordings.

we analyze the most obvious difference in recording conditions, namely the wideband and narrowband recordings. Figure 6-7 shows a heat map of the averaged posteriors for each dev set speaker in Cebuano generated by the LID DNN. As shown by the figure, for the majority of the speakers, Tagalog yields the highest posterior score. This makes sense because both the Cebuano and Tagalog corpora were recorded in the Philippines. Linguistically and acoustically (channel effects) they should be the most similar. However, we also notice that the wideband recordings from Cebuano prefers languages that also include wideband recordings, while the languages without wideband recordings get little to no posterior values. This is clear evidence that the LID DNN also takes into account the acoustics as well as the linguistics, which can be more preferable than just selecting the closest language based on linguistic knowledge. This heat map also points out the need for selecting just a portion of the data from a language, since the scores can vary greatly within a source language due to different recording conditions.

We then evaluate the averaged posteriors for each target language to identify the

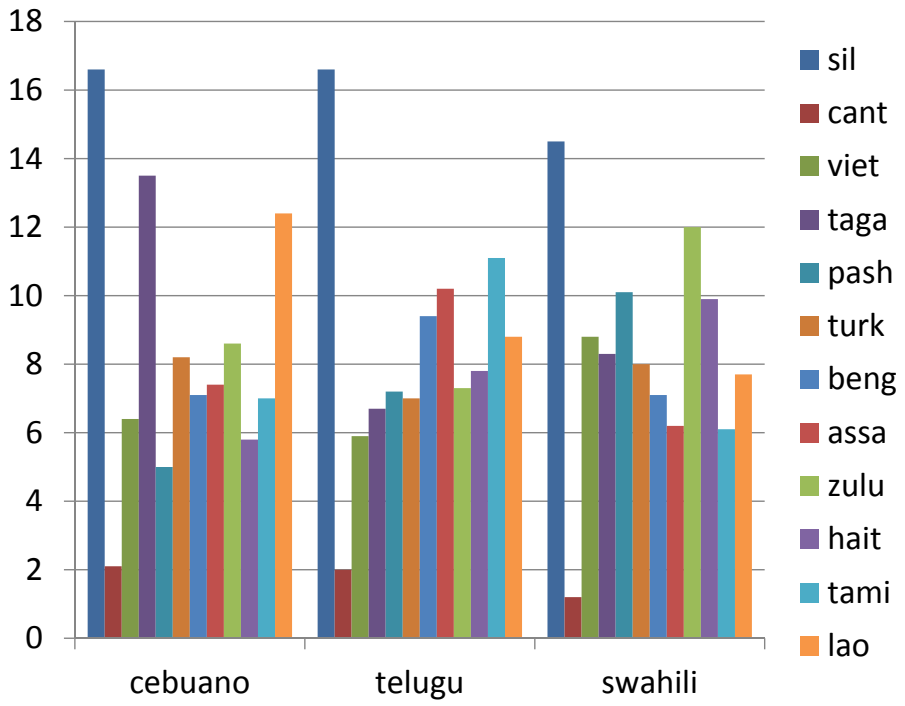

Figure 6-8: LID averaged posterior scores for each target language (in percent). Only the frames from narrowband utterances are used.

closest language, which we summarize in Fig. 6-8. To avoid the bias generated by the wideband recordings, we only use the narrowband portion to compute the average. Cebuano identifies Tagalog as the closest language followed by Lao. The top three for Telugu are Tamil, Assamese, and Bengali which are all Indian languages. Lastly, Swahili prefers Zulu. Thus, the LID DNN was able to identify the linguistically appropriate languages without any human knowledge. Note that in Section 6.3.2, we presented evidence that show a case where the LID DNN can identify a better language for cross language transfer learning than just pure linguistic knowledge.

6.4.4 Frame Selection DNN Analysis

We then analyze the posteriors generated by the frame selection DNNs. Figure 6-9 shows the posterior values averaged over all frames for each source-target pair. The overall rankings from the LID DNN and the frame selection DNN are similar. When Telugu is the target language, Assamese, Bengali, and Tamil still remain noticeably

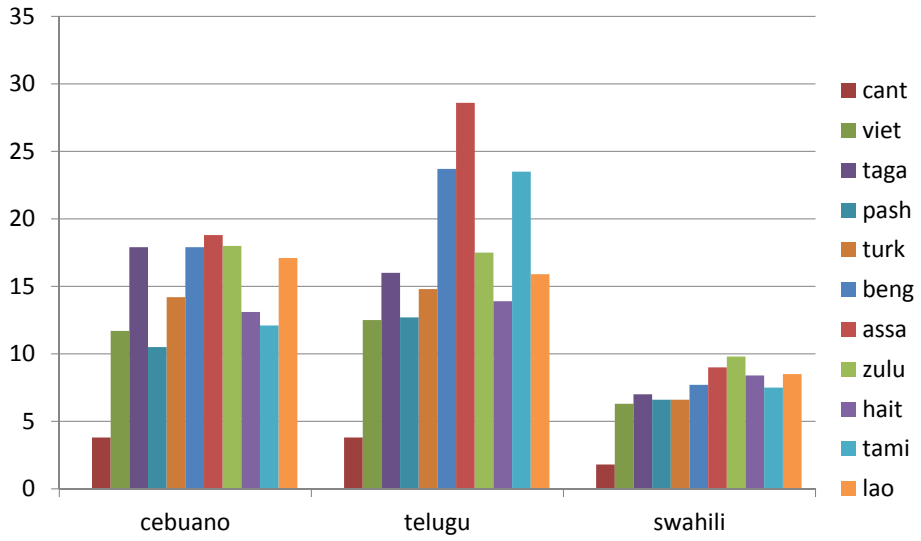

Figure 6-9: Probability of being the target language averaged over all frames of each source language. For each source-target pair, the posteriors are computed using the corresponding frame selection DNN. Values are shown in percent.

higher than the rest of the source languages. The highest match for Cebuano is now Assamese, but Tagalog follows closely behind. Lastly, Zulu is still favored by Swahili. However, the scores are lower compared to the other two target languages. This indicates that the source languages might not be as helpful for Swahili.

We also look into whether the LID DNN can select linguistically relevant frames. Figure 6-10 shows how many frames from the phonemes /s/ and /sh/ are selected from each source language (in %) when Cebuano is considered as the target language. Cebuano only has the phoneme /s/ but not /sh/. From the figure, we notice that /s/ frames are more likely to get selected over /sh/ for languages that have both /s/ and /sh/. This shows that the LID-based frame selection framework selects based on both acoustic and linguistic effects.

6.4.5 Recognition System

For each language, we used tied-state triphone CD-HMMs, with 2500 states and 18 Gaussian components per state. For the target languages we used a grapheme-based dictionary as described in Section 3.3. All the output targets of the SBN DNNs (including the multilingual SBN) were from CD states. We used the Block Softmax

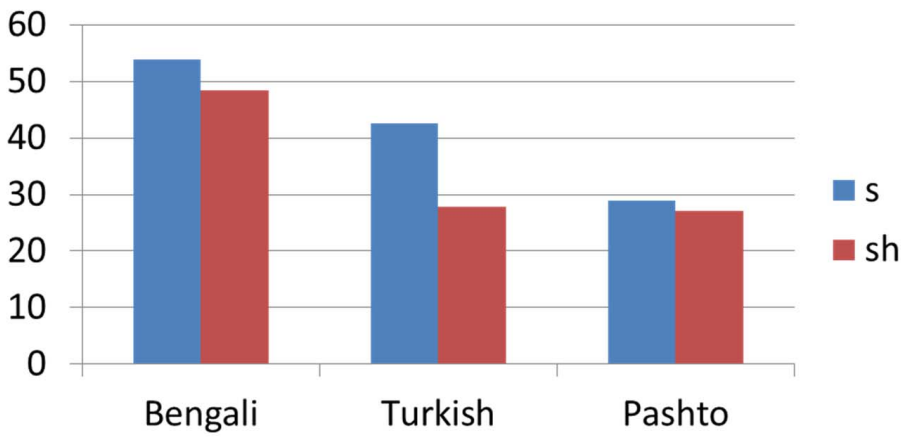

Figure 6-10: Percentage of frames from each phoneme selected from each source language for Cebuano is the target language.

method to train the multilingual SBN. We also kept only the SIL frames that appear 5 frames before and after actual speech. This reduced the total amount of frames for the multilingual DNN to around 520 hours. We observed no loss in accuracy from doing so, and it also reduced the training time significantly. Discriminative training was done on the CD-HMMs using the Minimum Bayes risk (MBR) criterion [23]. In this experiment, we also include web data as described in Section 3.5. The web data was cleaned and filtered using techniques described in [99]. For language modeling, n-gram LMs were created from training data transcripts and the web data. The LMs were then combined using weighted interpolation. The vocabulary included words that appeared in the training transcripts augmented with the top 30k most frequent words from the web.

6.4.6 Keyword Spotting

Keyword Spotting was done using a simplified version of what was described in Section 3.4. For this experiment, we report the KWS done using the development keywords on the 10 hour dev set. We did KWS on lattices using exact word matches, since we wanted to focus more on the difference in the recognizer. For this purpose, we only report the KWS numbers on in-vocabulary (IV) keywords only.

Table 6.7: ASR and KWS results. For MTWV, * indicates the value is significantly different from one in the row above (5% significance) using the Student's t-test.

Method	Cebuano		Telugu		Swahili	
	WER	MTWV	WER	MTWV	WER	MTWV
Monolingual SBN	73.5		86.4		65.8	
Adapted multilingual	65.0	0.2259	78.0	0.1269	54.9	0.3983
Closest language (mono re-train)	63.7	0.2526*	75.8	0.1682*	54.2	0.4225
100 hr closest frames	63.0	0.2513	76.0	0.1711*	52.4	0.4244*
200 hr closest frames	63.1	0.2531*	76.0	0.1756*	52.4	0.4233
All frames	63.0	0.2376*	75.8	0.1528*	52.4	0.4262*

6.4.7 Frame Selection Experiments

We compare the results between the three methods described earlier, namely a multilingual SBN adapted to the target language, a SBN trained by using the closest language as described in Section 6.3, and a SBN trained using frame selection. For frame selection, we have two configurations with 100 hours and 200 hours worth of closest frames. We did not go below 100 hours because there were too many class outputs that had no or too few frames. We also report the extreme situation where all 520 hours worth of frames are selected. Note that this is slightly different than the adapted multilingual SBN, since the second DNN for this case is trained on *adapted* BN features from the first BN. As a point of comparison, we also include a completely monolingual SBN trained only on the 3 hour VLLP data.

Table 6.7 summarizes the ASR and KWS results. As shown, monolingual SBNs perform significantly worse than multilingual techniques. This shows the strength of using multilingual data to help ASR in languages with limited resources. Using the closest source language to train the second DNN yields a noticeable improvement over the adapted multilingual SBN in both ASR and KWS. However, the gain in WER is smaller for Swahili. This can be attributed to the fact that the candidates for Swahili are worse than the other two languages, as noted in Section 6.4.4.

In terms of KWS, frame selection systems are significantly better than the ones using just the closest language. The best performance is achieved at 200 hours for

Cebuano and Telugu, and 100 hours for Swahili because Swahili has lower frame selection scores. The gain from frame selection over the closest language is higher in Telugu than in Cebuano. Telugu has more than one closest language so we expect more gain from using multiple languages. Frame selection scores for the case of Telugu are also higher than Cebuano signifying better synergies between the source and target language. Finally, using all frames performs worse than any kind of selection except for the case of Swahili where the frame selection scores are noticeably lower so the effect of having more data prevails. We believe that both the amount and the closeness of the data play a role in determining the benefits from multilingual training. Note that re-training 520 hours worth of frames is infeasible in a short time frame which might not be desirable for applications that require rapid deployment.

6.5 The Final Systems

Using all the techniques described in this thesis, we participated in the 2015 Babel OpenKWS Evaluation as part of the Babelon team. The target language for the evaluation was Swahili VLLP condition. We trained two speech recognizers as follows:

- Multilingual LrSBN system using LID-based adaptation

  This system is similar to the closest language system described in Section 6.4.7. However, we used the top 100,000 web words instead of the 30,000 used previously to push the OOV rate as low as possible since the main task is KWS. We did not use the frame selection method since re-training using the closest frames would take too long according the the evaluation rules which tries to limit the total development time to two weeks.

- Hybrid DNN-HMM Recurrent Neural Network system

  Recent advances in deep learning have improved the DNN architectures tremendously, making the hybrid DNN-HMM approach more competitive. Ideally, we would like to make use of the recent advances as well. To this end, used a novel model, the Prediction-Adaptation-Correction Recurrent Neural Network

(PACRNN), for acoustic modelling. PACRNN is a hybrid DNN-HMM system that has the ability to predict what you want to say ahead of time, adapts your listening effort according to the prediction, and finally do a correction once the information arrives [102]. Note that the methods we have described in this thesis, namely the BN features and LID-based transfer learning, can still be applied to hybrid architectures. The PACRNN was trained on BN features extracted from the first DNN of the adapted multilingual SBN. It was first trained on FLP Zulu (the closest language) and then adapted to Swahili using the same transfer learning techniques described in Section 6.3. With transfer learning, the PACRNN system improves by 1% absolute WER compared to training with just VLLP Swahili.

For each recognizer, we generated lattices based on three decoding units, words, morphemes, and syllables using the techniques described in Section 3.4.

6.5.1 Results

Table 6.8 summarizes the WER and MTWV results on dev and tune sets using their respective keyword lists. When considering whether a keyword is IV or OOV, we use the augmented vocabulary. PLP-word is a GMM-HMM system trained using PLP, F_0, and PoV features. The setup is otherwise the same as the SBN HMM-GMM system. However, in this setting, discriminative training failed to improve the recognition results, so we report the results when no discriminative training is applied. There is also no usage of web data. It is supposed to represent what happens when one attempts to build a traditional system on a low resource setup. The second baseline, Mono SBN-word, is a LrSBN system train on VLLP Swahili, e.g. a monolingual system. This represents a more modern baseline since it includes DNNs. Using the SBN features, discriminative training now improves the recognition results. Just like the PLP-word baseline, there is no usage of web data.

As shown from the table, the PACRNN systems which use the SBN HMM-GMM system as input features outperform the SBN systems in every case. This shows the

Table 6.8: ASR and KWS results on Swahili for different recognizers and decoding units.

System	tune WER	dev WER	tune MTWV			dev MTWV		
			IV	OOV	ALL	IV	OOV	ALL
SBN-word	57.8	52.9	0.3172	0.0000	0.2805	0.4494	0.0000	0.3885
SBN-morph			0.2915	0.1177	0.2701	0.3732	0.2411	0.3541
SBN-syl			0.2485	0.2512	0.2477	0.3389	0.4199	0.3495
PAC-word	55.1	49.5	0.4106	0.0000	0.3631	**0.4859**	0.0000	**0.4200**
PAC-morph			0.2899	0.1343	0.2702	0.4186	0.2558	0.3950
PAC-syl			0.3074	0.3711	0.3140	0.3754	**0.5186**	0.3946
All above						**0.5220**	**0.5454**	**0.5249**
PLP-word	71.5	68.2	0.1197	0.0000	0.1058	0.1924	0.0000	0.1663
Mono SBN-word	68.4	66.5	0.1410	0.0000	0.1246	0.2106	0.0000	0.1821

power of more sophisticated hybrid models over the SBN. However, the two set of systems combine very well as the system combination results (marked "All above") achieve 0.5249 MTWV, a 25% relative performance gain over our best single system. Using subword units, our performance on OOV keywords is even better than the IV words, showing the effectiveness of the subword methods.

Comparing against the monolingual baselines, SBN-word outperforms Mono SBN-word by more than twice in MTWV and 14 % absolute WER, which shows the power of multilingual training. Finally, the Mono SBN-word outperforms the PLP-word system by 0.02 in MTWV and 2% in WER, confirming the effectiveness of the SBN framework. Note that the evaluation goal was to exceed 0.3 MTWV, which all of our individual systems did, while none of the monolingual systems exceeded 0.2 MTWV.

OpenKWS 2015 Evaluation consists of two sponsored participants and several international teams of volunteers. To offer some perspective on our performance, our team, a sponsored participant, achieved 0.6006 ATWV on the dev set after combining 15 different systems, including ours. A system can have several decoding units, for example, we used two systems, the SBN and the PACRNN, with three decoding units, words, morphs, and syllables.

The best performing volunteer team obtained 0.5722 MTWV after system combi-

nation [7]. Cai *et al.* combined 11 systems, each with up to two decoding units, i.e. words and morphs. Their best performing single system achieved 0.4829 and 0.3570 MTWV for word- and morph-based recognizers, respectively. They also incorporate the proxy keyword technique [8] where words that sound similar to the OOV keywords are searched instead, so their word systems can also handle OOV keywords. Note that sponsored and volunteer participants have access to different sets of resources, so the numbers are not directly comparable. However, Cai *et al.* also used a variant of BN features trained on multilingual data which shows the effectiveness leveraging multilingual data.

6.6 Summary

In this chapter, we presented a method to investigated the use of transfer learning for low resource language ASR. To get the best performance it is important to identify the source language that is most similar to the target language. We proposed a framework that can be used to measure language similarity in a data-driven manner by using DNNs. This method not only is able to identify language based on linguistic closeness, but it also can take into account of the acoustic conditions of the recordings.

We also investigated the best method in adapting a multilingual SBN to any target language. We found using the closest language, identified by the LID, to initialize the second DNN to work best. This method can also be improved even further by selecting frames from the entire pool of source languages.

The LID-based transfer learning technique is not limited to just Tandem methods, it also applies to hybrid DNN-HMM models which ultimately produce the best results.

Chapter 7

Conclusion

7.1 Summary

In this thesis, we took on the challenge of low resource ASR for the applications of automatic transcription and KWS. The research is motivated by the fact that most languages of the world are still lacking the ASR capabilities while the need for ASR has grown ever larger with the spread of mobile and personal recording devices. The three building blocks of ASR, the acoustic model, the lexicon, and the language model, all require resources on the order that might not be available for most languages.

We investigated techniques that helped these three components. To deal with the need for a lexicon, a graphemic lexicon can be used to completely ignore the issue with some small degradation in performance. If a small starting lexicon exists, techniques such as the Pronunciation Mixture Model (PMM) can generate reasonable lexicons in a data-driven manner. For the language model, we have investigated the use of web data and subword units for improving KWS performance. Although web data can sometimes hurt transcription performance due to the style mismatch, it helps improve KWS by reducing the number of OOV keywords.

For better acoustic models, we proposed a method to learn better feature representations using the Low-rank Stacked Bottleneck Network (LrSBN). This method, albeit developed for the use of low resource ASR in mind, also helped improve ASR performance in larger tasks. Using the LrSBN we applied multilingual techniques via

the use of transfer learning which helped alleviate the lack of acoustic data. We also proposed a method to select the source language for transfer learning by using a DNN trained for Language Identification. The data selection can also be done on the frame level to help improve the performance even further. Using techniques described and proposed in this thesis, we were able to more than double the KWS spotting performance for low resource language compared to using standard techniques geared towards rich resource domains.

7.2 Future Work and Directions

There are short term improvements that can be done with regards to the LrSBN model. One is to investigate more sophisticated DNN architectures such as Convolutional Neural Networks (CNNs) to help handle the variability due to the vocal tract. More investigation also needs to be done in order to make the frame selection framework to work in a scenario with a tight time constraint. One solution might be to use parallel training methods instead of using a single GPU.

Since LrSBN is, in essence, a feature extractor, it can in theory be used for other speech related task such as Speaker Identification or Language Identification. There is some preliminary exploration on Language Identification showing promising results [17]. It is also worth investigating to see if such a multi-task training approach can be used to extract features for auditory scene analysis, the task of analysing what is happening in an audio recording.

There are also directions of potential research that this thesis does not address but are important for the development of low resource ASR.

7.2.1 Handling Dialects and Accented Speech

In this thesis, we completely ignored the dialect variants in each language. ASR performance on the Babel corpus can vary greatly between dialects depending on which dialect dominates the training data. A possible venue for exploration includes having dialect specific models that are adapted from the dialect-independent model (just

like how we trained model for the target language using a more general multilingual model). Another related issue, is accented speech recognition. Multilingual ASR can potentially improves the performance of accented speech due to its ability to model the characteristics of more than just a single language.

7.2.2 ASR with Zero Transcription

All work done in this thesis still relies on some amount of transcribed speech. An even more challenging task is what can one do with just speech data. Work in [45] had attempted this using the Babel corpus, but the performance is far from usable even from the KWS point of view. They also assumed the existence of a lexicon in the target language, an assumption that might not hold true. A related issue with zero transcription and no lexicon is how can one relate letters of the language with sounds of the language, since the two are no longer in parallel, a task of decipherment.

7.2.3 Mis-match Crowd-Sourcing for ASR

In general, finding recordings of a low resource language is easier than finding transcribed recordings. When the language has few speakers it is often hard to find transcribers even when the recordings are plentiful. This is especially true in the Internet era. One possible solution to this problem is to have a non-speaker of the language transcribe the recordings into sounds or words of his own language. The task, called mis-match crowd-sourcing, has been explored in [39, 52]. However, the exploration is still preliminary and mostly done on the phoneme level rather than as a Large Vocabulary Continuous Speech Recognition (LVCSR) task. The crowd-sourced transcription can be used to augment the available transcribed data. Again, an existing phonetic lexicon in the target language is assumed. Otherwise, this is also a decipherment task if there are no zero transcription by speakers of the language.

7.2.4 ASR for Languages without a Writing System

Every language in the Babel corpus has a writing system that are more or less standardized. In some languages such as Hokkien, there is no standardized writing system (even the set of characters/alphabets may not be standardized). In this case, the ASR system needs to be able to handle noise in the transcription. There are also languages with no writing systems at all, for example the Wu dialect of Chinese. Can one make an ASR system which generates no tangible output? A solution might be to tie ASR with specific tasks that requires no written form. Possible applications are Query by example related applications, translation, and dialogue systems. Query-by-example is a task where given a speech input, find all the occurrences of that term in the database of recordings. The search target might not necessary be a spoken document. One can say "give me pictures of oranges" and the system should show pictures of oranges. One such system is already being explored in [32].

7.2.5 Multilingual Techniques for Language Modeling

The sharing of multilingual resources in this thesis are primarily on acoustic data. Lamel *et al.*. had explored the use of machine translation for augmenting the text used for language modeling [25]. However, it is often hard to find parallel text especially for low-resource languages. Just like how the sounds of different language can be similar, there are also strong correspondence between languages in syntax and grammar. One example is the work by Snyder *et al.* which used a multilingual corpora to better learn morphological segmentations [83]. One would wonder whether multilingual data sharing can be used to improve the language model in a data-driven manner with little linguistic knowledge.

7.3 Closing Statement

In closing, it is worth remembering some of the original motivation for this work. There are around 7,000 languages in the world. In this thesis, we have presented

work conducted on 18 languages. Although the techniques presented in this work are not specific to any particular language, there is still much work to be done to reach the full coverage of all the world's languages. We hope that the work conducted in this thesis will help the effort in diffusing ASR technologies to everyone across the globe. Many of the languages now only have just a handful of speakers, which might once again requires another shift in the ASR paradigm.

Glossary of Acronyms

F_0 Fundamental Frequency. 58

Δ Delta features. 38

Δ^2 Delta-delta features. 38

ALP Active Learning Language Pack. 54

AM Acoustic Model. 31

ASR Automatic Speech Recogition. 31

ATWV Averaged Term Weighted Value. 40

BN Bottleneck. 47

CD Context Dependent. 33

CE Cross Entropy. 45

CER Character Error Rate. 35

CI Context Independent. 33

CMLLR Constrained Maximum Likelihood Linear Regression. 39

CMVN Cepstral Mean and Variance Normalization. 37

FLP Full Language Pack (40-80 hours). 54

fMLLR Feature-Space Maximum Likelihood Linear Regression. 39

GMM Gaussian Mixture Model. 32

HMM Hidden Markov Model. 32

IPA International Phonetic Alphabet. 72

IV In-Vocabulary. 34

KWS Keyword Spotting. 39

LDA Linear Discriminant Analysis. 38

LID Language Identification. 92

LLP Limited Language Pack (10 hours). 54

LM Language Model. 34

LPC Linear Predictive Coding. 35

LrSBN Low-rank Stacked Bottleneck. 79

MFCC Mel-frequency Cepstral Coefficients. 31

ML Maximum Likelihood. 33

MLLT Maximum Likelihood Linear Transform. 39

MMI Maximum Mutual Information. 33

MPE Minimum Phone Error. 33

MTWV Maximum Term Weighted Value. 41

OOV Out-of-Vocabulary. 34

PCA Principle Component Analysis. 38

PLP Perceptual Linear Prediction. 35

PoV Probability of Voicing. 58

ReLU Rectified Linear Unit. 43

RNN Recurrent Neural Network. 34

SAMPA Speech Assessment Methods Phonetic Alphabet. 72

SBN Stacked Bottleneck. 78

SGD Stochastic Gradient Descent. 44

sMBR state-level Minimum Bayes Risk. 33

STFT Short-Time Fourier Transform. 36

VLLP Very Limited Language Pack (3 hours). 54

VTLN Vocal Tract Length Normalization. 38

WER Word Error Rate. 35

Appendix A

Global phone mappings

In this segment, we present the phoneme mappings we used for the experiments in Chapter 6.3. There are no extra mappings for Pashto.

Table A.1: Global phoneset.

	phone	description
1	p	unvoiced labial stop
2	t	unvoiced dental stop
3	c	unvoiced palatal stop
4	k	unvoiced velar stop
5	q	unvoiced uvular stop
6	b	voiced labial stop
7	d	voiced dental stop
8	g	voiced velar stop
9	t‘	unvoiced retroflex stop
10	d‘	voiced retroflex stop
11	?	glottal stop
12	ts	unvoiced alveolar affricate
13	tS	unvoiced palato-alveolar affricate
14	dz	unvoiced alveolar sibilant affricate
15	dZ	voiced palato-alveolar affricate
16	f	unvoiced labial fricative
17	s	unvoiced alveolar fricative
18	S	unvoiced palato-alveolar fricative
19	C	unvoiced palatal fricative
20	x	unvoiced velar fricative
21	T	unvoiced dental fricative
22	h	glottal fricative
23	v	voiced labio-dental fricative
24	z	voiced alveolar fricative
25	Z	voiced palato-alveolar fricative
26	G	voiced velar fricative
27	j\	voiced palatal fricative
28	s‘	unvoiced retroflex fricative
29	z‘	voiced retroflex fricative
30	l	alveolar lateral approximant
31	w	labio-velar approximant
32	j	voiced palatal approximant
33	5	velarized alveolar lateral approximant
34	r	alveolar approximant

Table A.2: Global phoneset (cont.).

	phone	description
35	m	labial nasal
36	n	alveolar nasal
37	N	velar nasal
38	n‘	palatal nasal
39	3	unrounded open-mid central
40	4	rhotic alveolar
41	i	unrounded high front
42	i:	long unrounded high front
43	I	unrounded near-close near-front
44	I:	long unrounded near-close near-front
45	y	rounded high front
46	y:	long high front
47	1	unrounded high back
48	1:	long unrounded high back
49	u	rounded high back
50	u:	long unrounded high back
51	e	unrounded close-mid front
52	e:	long unrounded close-mid front
53	2	rounded close-mid front
54	2:	long rounded close-mid front
55	@	schwa
56	o	rounded close-mid back
57	o:	long rounded close-mid back
58	E	unrounded open-mid front
59	E:	long unrounded open-mid front
60	9:	long rounded close-mid central
61	O	rounded open-mid back
62	O:	long rounded open-mid back
63	a	unrounded low front
64	a:	long unrounded low front
65	6	near-open central
66	V	unrounded open-mid back
67	A	unrounded low back

Table A.3: Extra mappings for Cantonese.

original	global phoneset
6j	6 j
6w	6 w
9y	9 y
O:j	O: j
a:j	a: j
a:w	a: w
ej	e j
gw	g w
iw	i w
kw	k w
ow	o w
u:j	u: j

Table A.4: Extra mappings for Turkish.

original	global phoneset
gj	j
r\	r

Table A.5: Extra mapping for Turkish.

original	global phoneset
gj	g
r\	r

Table A.6: Extra mappings for Tagalog.

original	global phoneset
ae	a e
aj	a j
aw	a w
oj	o j

Table A.7: Extra mappings for Vietnamese.

original	global phoneset
1@I	1 @ I
1@	1 @
1@U	1 @ U
1U	1 U
a:I	a: U
aU	a U
b_<	b
d_<	d
eU	e U
EU	E U
i@	i @
@:I	@ I
@I	@ I
@:	@
i@U	i @ U
iU	i U
J\	z
oaI	o a I
oaI:	o a I:
Oa	O a
Oa:	O aa
OE	O E
oI	o I
OI	O I
r\	r
t_h	t
ts‘	t‘
ue	u e
ui:	u i:
u@I	u @ I
uI	u I
uI@	u I at
u@	u @
u@:	u @
@U	@ U

Appendix B

Babel data

The version of the language packs used in this thesis are shown in Table B.1.

Table B.1: Language pack version for each language.

Language	Version
Cantonese	IARPA-babel101-v0.4c
Assamese	IARPA-babel103b-v0.3
Bengali	IARPA-babel102b-v0.4
Pashto	IARPA-babel104b-v0.4aY
Turkish	IARPA-babel105b-v0.4
Tagalog	IARPA-babel106-v0.2g
Vietnamese	IARPA-babel107b-v0.7
Haitian	IARPA-babel201b-v0.2b
Swahili	IARPA-babel202b-v1.0d-build
Lao	IARPA-babel203b-v2.1a
Tamil	IARPA-babel204b-v1.1b
Kurmanji	IARPA-babel205b-v1.0a
Zulu	IARPA-babel206b-v0.1e
Tok Pisin	IARPA-babel207b-v1.0e-build
Cebuano	IARPA-babel301b-v2.0b
Kazakh	IARPA-babel302b-v1.0a
Telugu	IARPA-babel303b-v1.0a
Lithuanian	IARPA-babel304b-v1.0b
Guarani	IARPA-babel305b-v1.0b
Igbo	IARPA-babel306b-v2.0c
Amharic	IARPA-babel307b-v1.0b
Mongolian	IARPA-babel401b-v2.0b
Javanese	IARPA-babel402b-v1.0b
Dholuo	IARPA-babel403b-v1.0b

Bibliography

[1] L. Bahl, P. F. Brown, P. V. De Souza, and R. L. Mercer. Maximum mutual information estimation of hidden markov model parameters for speech recognition. In *Proc. ICASSP*, 1986. 33

[2] T. Bazillon, Y. Esteve, and D. Luzzati. Manual vs assisted transcription of prepared and spontaneous speech. In *Proc. LREC*, 2008. 24

[3] M. Bisani and H. Ney. Joint-sequence models for grapheme-to-phoneme conversion. *Speech Communication*, 50:434 – 451, 2008. 61

[4] C. M. Bishop. *Neural networks for pattern recognition*. Oxford university press, 1995. 38

[5] P. Boersma. Accurate short-term analysis of the fundamental frequency and the harmonics-to-noise ratio of a sampled sound. In *Proc. the Institute of Phonetic Sciences*, 1993. 58

[6] L. Burget, P. Schwarz, M. Agarwal, et al. Multilingual acoustic modeling for speech recognition based on subspace gaussian mixture models. In *Proc. ICASSP*, 2010. 27, 71

[7] M. Cai, Z. Lv, C. Lu, J. Kang, L. Hui, Z. Zhang, and J. Liu. High-performance Swahili keyword search with very limited language pack: The THUEE system for the OpenKWS15 evaluation. In *Proc. ASRU*, 2015. 114

[8] G. Chen, O. Yilmaz, J. Trmal, D. Povey, and S. Khudanpur. Using proxies for OOV keywords in the keyword search task. In *Proc. ASRU*, 2013. 66, 114

[9] G. Chen, D. Povey, and S. Khudanpur. Acoustic data-driven pronunciation lexicon generation for logographic languages. In *Proc. ICASSP*, 2016. 62

[10] E. Chuangsuwanich, Y. Zhang, and J. Glass. Multilingual data selection for training stacked bottleneck features. In *Proc. InterSpeech*, 2016. 92

[11] J. Cohen, T. Kamm, and A. G. Andreou. Vocal tract normalization in speech recognition: Compensating for systematic speaker variability. *The Journal of the Acoustical Society of America*, 97(5):3246–3247, 1995. 38

[12] N. Dave. Feature extraction methods LPC, PLP and MFCC in speech recognition. *International Journal for Advance Research in Engineering and Technology*, 1(6):1–4, 2013. 31

[13] M. Davel, E. Barnard, C. van Heerden, W. Hartmann, D. Karakos, R. Schwartz, and S. Tsakalidis. Exploring minimal pronunciation modeling for low resource languages. In *Proc. InterSpeech*, 2015. 65

[14] V. Doumpiotis, S. Tsakalidis, and W. Byrne. Discriminative training for segmental minimum bayes risk decoding. In *Proc. ICASSP*, 2003. 33

[15] W. Ellermeier and G. Faulhammer. Empirical evaluation of axioms fundamental to stevenss ratio-scaling approach: I. loudness production. *Perception & Psychophysics*, 62(8):1505–1511, 2000. 36

[16] D. Ellis. ICSI speech FAQ: 6.3 how are neural nets trained?, Aug 2000. URL http://www1.icsi.berkeley.edu/speech/faq/nn-train.html. 46

[17] R. Fér, P. Matějka, F. Grézl, O. Plchot, and J. Černockỳ. Multilingual bottleneck features for language recognition. In *InterSpeech*, 2015. 116

[18] J. Fiscus, J. Ajot, J. S. Garofolo, and G. Doddingtion. Results of the 2006 spoken term detection evaluation. In *Proc. ACM SIGIR Workshop on Searching Spontaneous Conversational Speech*, 2007. 40

[19] T. Fraga-Silva, J.-L. Gauvain, L. Lamel, A. Laurent, V.-B. Le, A. Messaoudi, V. Vapnarsky, C. Barras, C. Becquey, D. Doukhan, et al. Active learning based data selection for limited resource STT and KWS. In *Proc. InterSpeech*, 2015. 15, 25, 55

[20] M. J. F. Gales. Maximum likelihood linear transformation for HMM-based speech recognition. In *Comp.Speech & Language*, 1998. 80, 86

[21] M. J. F. Gales. Semi-tied covariance matrices for hidden markov models. In *IEEE Trans. on Speech and Audio*, 1999. 39, 80

[22] P. Ghahremani, B. BabaAli, K. R. D. Povey, J. Trmal, and S. Khudanpur. A pitch extraction algorithm tuned for automatic speech recognition. In *Proc. ICASSP*, 2014. 58

[23] M. Gibson and T. Hain. Hypothesis spaces for minimum bayes risk training in large vocabulary speech recognition. In *Proc. InterSpeech*, 2006. 84, 92, 109

[24] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proc. International conference on artificial intelligence and statistics*, 2010. 46

[25] A. Gorin, R. Lileikyte, G. Huang, L. Lamel, J.-L. Gauvain, and A. Laurent. Language model data augmentation for keyword spotting in low-resourced training conditions. In *Proc. Interspeech 2016*, In Review. 118

[26] F. Grézl and M. Karafiát. Adapting multilingual neural network hierarchy to a new language. In *Proc. SLTU*, 2014. 28

[27] F. Grézl, M. Karafiát, S. Kontár, and J. Černocký. Probabilistic and bottle-neck features for LVCSR of meetings. In *Proc. ICASSP*, 2007. 47

[28] F. Grézl, M. Karafiát, and M. Janda. Study of probabilistic and bottle-neck features in multilingual environment. In *Proc. ASRU*, 2011. 75, 86, 87

[29] F. Grézl, E. Egorova, and M. Karafiát. Further investigation into multilingual training and adaptation of stacked bottle-neck neural network structure. In *Proc. SLT*, 2014. 28, 103

[30] F. Grézl, M. Karafiát, and K. Veselý. Adaptation of multilingual stacked bottle-neck neural network structure for new languages. In *Proc. ICASSP*, 2014. 89, 91, 97

[31] D. Harwath and J. Glass. Speech recognition without a lexicon-bridging the gap between graphemic and phonetic systems. In *Proc. InterSpeech*, 2014. 60

[32] D. Harwath and J. Glass. Deep multimodal semantic embeddings for speech and images. In *Proc. ASRU*, 2015. 118

[33] H. Hermansky. Perceptual linear predictive (PLP) analysis of speech. *the Journal of the Acoustical Society of America*, 87(4):1738–1752, 1990. 35

[34] J. L. Hieronymus. ASCII phonetic symbols for the worlds languages: Worldbet. *Journal of the International Phonetic Association*, 23, 1993. 72

[35] G. Hinton, L. Deng, D. Yu, et al. Deep neural networks for acoustic modeling in speech recognition. In *IEEE Signal Processing Magazine*, volume 28, pages 82–97, November 2012. 27, 42, 44, 47

[36] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006. 46

[37] X. Huang, A. Acero, and H.-W. Hon. *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development*. Prentice Hall PTR, 1st edition, 2001. 36

[38] International Phonetic Association. *Handbook of the International Phonetic Association: A guide to the use of the International Phonetic Alphabet*. Cambridge University Press, 1999. 72

[39] P. Jyothi and M. Hasegawa-Johnson. Transcribing continuous speech using mismatched crowdsourcing. In *Proc. InterSpeech*, 2015. 117

[40] M. Karafiát and F. Grézl. Hierarchical neural net architectures for feature extraction in ASR. In *Proc. InterSpeech*, 2010. 79

[41] M. Karafiát, F. Grézl, M. Hannemann, K. Veselý, and J. H. Černocký. BUT Babel system for spontaneous cantonese. In *Proc. InterSpeech*, 2013. 61, 78, 81, 82

[42] D. Karakos, R. Schwartz, S. Tsakalidis, L. Zhang, S. Ranjan, T. Ng, R. Hsiao, G. Saikumar, I. Bulyko, L. Nguyen, J. Makhoul, F. Grezl, M. Hannemann, M. Karafiat, I. Szoke, K. Vesely, L. Lamel, and V.-B. Le. Score normalization and system combination for improved keyword spotting. In *Proc. ASRU*, 2013. 41

[43] K. Kirchhoff and D. Vergyri. Cross-dialectal data sharing for acoustic modeling in Arabic speech recognition. *Speech Communication*, 46(1):37–51, 2005. 15, 25

[44] K. Knill, M. Gales, S. Rath, P. Woodland, C. Zhang, and S. Zhang. Investigation of multilingual deep neural networks for spoken term detection. In *Proc. ASRU*, 2013. 86, 103

[45] K. Knill, M. J. Gales, S. P. Rath, P. C. Woodland, C. Zhang, and S.-X. Zhang. Investigation of multilingual deep neural networks for spoken term detection. In *Proc. ASRU*, 2013. 76, 89, 117

[46] G. Kumar, M. Post, D. Povey, and S. Khudanpur. Some insights from translating conversational telephone speech. In *Proc. ICASSP*, 2014. 15, 24, 25

[47] V. Le, L. Lamel, A. Messaoudi, W. Hartmann, J. Gauvain, C. Woehrling, J. Despres, and A. Roy. Developing STT and KWS systems using limited language resources. In *Proc. InterSpeech*, 2014. 60

[48] H. Lee, Y. Zhang, E. Chuangsuwanich, and J. Glass. Graph-based re-ranking using acoustic feature similarity between search results for spoken term detection on low-resource languages. In *Proc. InterSpeech*, 2014. 64

[49] G. F. S. Lewis, M. Paul and C. D. Fennig. *Ethnologue: Languages of the World, Nineteenth edition.* SIL International. Online version: http://www.ethnologue.com, 2016. 26

[50] X. Li and X. Wu. Constructing long short-term memory based deep recurrent neural networks for large vocabulary speech recognition. In *Proc. ICASSP*, 2015. 15, 24, 25

[51] R. P. Lippmann. Speech recognition by machines and humans. *Speech communication*, 22(1):1–15, January 1997. 24

[52] C. Liu, P. Jyothi, H. Tang, V. Manohar, M. Hasegawa-Johnson, and S. Khudanpur. Adapting ASR for under-resourced languages using mismatched transcriptions. In *Proc. ICASSP*, 2016. 117

[53] A. L. Maas, P. Qi, Z. Xie, A. Y. Hannun, C. T. Lengerich, D. Jurafsky, and A. Y. Ng. Building DNN acoustic models for large vocabulary speech recognition. *arXiv preprint arXiv:1406.7806*, 2014. 15, 25

[54] J. Makhoul. Spectral linear prediction: properties and applications. *IEEE Trans. on Acoustics, Speech and Signal Processing*, 23(3):283–296, 1975. 36

[55] J. Mamou, J. Cui, X. Cui, M. J. Gales, B. Kingsbury, K. Knill, L. Mangu, D. Nolden, M. Picheny, B. Ramabhadran, et al. System combination and score normalization for spoken term detection. In *Proc. ICASSP*, 2013. 41

[56] I. McGraw, I. Badr, and J. Glass. Learning lexicons from speech using a pronunciation mixture model. *IEEE Trans. on Acoustics, Speech, and Signal Processing*, 2012. 60

[57] T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur. Recurrent neural network based language model. In *Proc. InterSpeech*, 2010. 34

[58] A. Mohan and R. Rose. Multi-lingual speech recognition with low-rank multi-task deep neural networks. In *Proc. ICASSP*, 2015. 89

[59] B. C. J. Moore. *An Introduction to the Psychology of Hearing*. Academic Press, 1997. 36

[60] K. Narasimhan, D. Karakos, R. Schwartz, S. Tsakalidis, and R. Barzilay. Morphological segmentation for keyword spotting. In *Proc. EMNLP*, 2014. 63, 65

[61] D. Povey. *Discriminative training for large vocabulary speech recognition*. PhD thesis, University of Cambridge, 2005. 33

[62] D. Povey and B. Kingsbury. Evaluation of proposed modifications to MPE for large scale discriminative training. In *Proc. ICASSP*, 2007. 33

[63] D. Povey and K. Yao. A basis method for robust estimation of constrained MLLR. In *Proc. ICASSP*, 2011. 39

[64] D. Povey, B. Kingsbury, L. Mangu, et al. fMPE: Discriminatively trained features for speech recognition. In *Proc. ICASSP*, 2005. 85

[65] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlíček, Y. Qian, P. Schwarz, J. Silovský, G. Stemmer, and K. Veselý. The Kaldi speech recognition toolkit. In *Proc. ASRU*, 2011. 58, 80, 81

[66] K. Precoda. Non-mainstream languages and speech recognition: some challenges. *CALICO Journal*, 21(2), January 2004. 23

[67] K. Probst. *Learning transfer rules for machine translation with limited data*. PhD thesis, Carnegie Mellon University, 2005. 26

[68] L. Rabiner and B.-H. Juang. *Fundamentals of speech recognition*. Prentice hall, 1993. 33

[69] A. Ragni, M. Gales, and K. Knill. A language space representation for speech recognition. In *Proc. ICASSP*, 2015. 27

[70] A. Ragni, M. Gales, and K. Knill. A language space representation for speech recognition. In *Proc. ICASSP*, 2015. 75, 105

[71] S. P. Rath, D. Povey, K. Veselý, and J. H. Černocký. Improved feature processing for deep neural networks. In *Proc. InterSpeech*, 2013. 81

[72] P. S. Ray, M. A. Hai, and L. Ray. *Bengali Language Handbook*, chapter 1. Center for Applied Linguistics, 1966. 92

[73] R. Rosenfeld. Two decades of statistical language modeling: Where do we go from here? In *Proc. of the IEEE*, volume 88, 2000. 34

[74] R. Sahraeian and D. Van Compernolle. A study of rank-constrained multilingual DNNs for low-resource ASR. In *Pro. ICASSP*, 2016. 89

[75] T. N. Sainath, B. Kingsbury, and B. Ramabhadran. Auto-encoder bottleneck features using deep belief networks. In *Proc. ICASSP*, 2012. 27

[76] T. N. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, and B. Ramabhadran. Low-rank matrix factorization for deep neural network training with high-dimensional output targets. In *Proc. ICASSP*, 2013. 78, 79, 83

[77] T. N. Sainath, A. Mohamed, B. Kingsbury, and B. Ramabhadran. Deep convolutional neural networks for lvcsr. In *Proc. ICASSP*, 2013. 97

[78] G. Saon, H.-K. J. Kuo, S. Rennie, and M. Picheny. The IBM 2015 English conversational telephone speech recognition system. *arXiv preprint arXiv:1505.05899*, 2015. 15, 24, 25, 26

[79] T. Schultz. GlobalPhone: a multilingual speech and text database developed at Karlsruhe University. In *Proc. InterSpeech*, 2002. 71

[80] T. Schultz and K. Kirchhoff. *Multilingual Speech Processing*. Elsevier, 2006. 27

[81] T. Schultz and A. Waibel. Language independent and language adaptive large vocabulary speech recognition. In *Proc. ICSLP*, 1998. 71, 74

[82] F. Seide, H. Fu, J. Droppo, G. Li, and D. Yu. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech DNNs. In *Proc. InterSpeech*, 2014. 15, 24, 25

[83] B. Snyder and R. Barzilay. Unsupervised multilingual learning for morphological segmentation. In *Proc. ACL*, 2008. 118

[84] H. Soltau, G. Saon, and T. N. Sainath. Joint training of convolutional and non-convolutional neural networks. In *Proc. ICASSP*, 2014. 15, 24, 25

[85] A. Stolcke, F. Grézl, M. Hwang, et al. Cross-domain and cross-language portability of acoustic features estimated by multilayer perceptrons. In *Proc. ICASSP*, 2006. 27, 91

[86] H. Su and H. Chen. Experiments on parallel training of deep neural network using model averaging. *arXiv preprint arXiv:1507.01239*, 2015. 15, 24, 25

[87] H. Su, G. Li, D. Yu, and F. Seide. Error back propagation for sequence training of context-dependent deep networks for conversational speech transcription. In *Proc. ICASSP*, pages 6664–6668, 2013. 45

[88] D. Talkin. A robust algorithm for pitch tracking (RAPT). *Speech coding and synthesis*, 495:518, 1995. 58

[89] D. Talkin. *A Robust Algorithm for Pitch Tracking*, chapter 4. Speech Coding and Synthesis, 2013. 80

[90] S. Tsakalidis, X. Zhuang, R. Hsiao, S. Wu, P. Natarajan, R. Prasad, and P. Natarajan. Robust event detection from spoken content in consumer domain videos. In *Proc. InterSpeech*, 2012. 52

[91] F. Valente, J. Vepa, C.Plahl, et al. Hierarchical neural networks feature extraction for LVCSR system. In *Proc. InterSpeech*, 2007. 78

[92] K. Veselý, M. Karafiát, and F. Grézl. Convolutive bottleneck network features for LVCSR. In *Proc. ASRU*, 2011. 27

[93] K. Veselý, M. Karafiát, F. Grézl, et al. The language-independent bottleneck features. In *Proc. SLT*, 2012. 27, 87, 91

[94] K. Veselý, A. Ghoshal, and D. Povey. Sequence-discriminative training of deep neural networks. In *Proc. InterSpeech*, 2013. 33, 81

[95] N. T. Vu, F. Metze, and T. Schultz. Multilingual bottle-neck features and its application for under-resourced languages. In *Proc. SLT*, 2012. 27, 79, 91

[96] J. C. Wells et al. SAMPA computer readable phonetic alphabet. *Handbook of standards and resources for spoken language systems*, 4, 1997. 72

[97] Z. J. Yan, Q. Huo, and J. Xu. A scalable approach to using DNN-derived features in GMM-HMM based acoustic modeling for LVCSR. In *Proc. InterSpeech*, 2013. 27

[98] D. Yu and M. L. Seltzer. Improved bottleneck features using pretrained deep neural networks. In *Proc. InterSpeech*, 2011. 27, 82

[99] L. Zhang, D. Karakos, W. Hartmann, R. Hsiao, R. Schwartz, and S. Tsakalidis. Enhancing low resource keyword spotting with automatically retrieved web documents. In *Proc. InterSpeech*, 2015. 66, 109

[100] Y. Zhang, E. Chuangsuwanich, and J. Glass. Language ID-based training of multilingual stacked bottleneck features. In *Proc. InterSpeech*, 2014. 92, 95

[101] Y. Zhang, E. Chuangsuwanich, and J. Glass. Extracting deep neural network bottleneck features using low-rank matrix factorization. In *Proc. ICASSP*, 2014. 17, 77, 80

[102] Y. Zhang, E. Chuangsuwanich, J. Glass, and D. Yu. Prediction-Adaptation-Correction recurrent neural networks for low-resource language speech recognition. In *Proc. ICASSP*, 2016. 112